目 录

第1章 MATLAB仿真技术与应用简介

知识点：

- 系统仿真技术概述
- MATLAB仿真技术的发展与应用
- MATLAB仿真技术的特点
- 仿真应用实例简介

本章主要介绍了MATLAB系统仿真技术的相关概念与发展历程，MATLAB仿真技术的简单应用与它的功能、特点，及其应用于系统仿真时所具有的得天独厚的优势。同时还介绍了其动态仿真工具包SIMULINK的特色。最后通过应用实例说明MATLAB/SIMULINK的功能与特点以及在Internet上丰富的MATLAB资源。

通过本章的学习，读者能够从总体上了解MATLAB系统仿真技术及其应用，对学习后续章节的知识将有很大的帮助。

1.1 系统仿真技术概述

在进行系统仿真的过程中，系统是系统分析研究的对象，模型是系统分析中能够表现系统本质的一种描述。此外，模型也是系统仿真、分析、控制和优化的基础。而系统仿真技术则是建立在模型基础上的一种实验方法。理解了系统、模型等基本概念之后才能够全面而准确地理解系统仿真技术。

系统是指具有某种特定功能，按照一定规律结合起来的相互作用又相互联系的对象的集合。建立系统概念的目的是为了能够更加深入地了解并掌握系统的特性与变化规律。系统通常包括实体、属性和行为三个方面的内容。实体（也可以称为对象）是指组成系统的具体的单元；属性则用于描述系统中各个实体的特性；行为是指实体随着时间变化面发生的状态变化。比如一个控制系统包括被控对象、控制器以及与之相关的各种信号（主要包括控制信号、参考输入信号、被控信号和干扰信号等），其中，被控对象和控制器是系统的实体。

系统可以分为工程系统和非工程系统。工程系统主要包括电气、机械、通信等工程应用领域的系统；非工程系统的范围则更加广泛，如社会系统、经济系统、生态系统、管理系统等。此外，系统根据系统的状态随时间变化情况可以分为连续系统和离散系统。系统的状态随时间连续变化的系统称为连续系统，可以用微分方程描述连续系统的属性。系统的状态变化发生在离散时刻的系统称为离散系统。离散系统又可分为离散时间系统和离散事件系统。离散时间系统是目前应用较为广泛的系统，如数控系统就是典型的离散

时间系统。离散时间系统的属性一般用差分方程来描述。离散事件系统是一种随机事件系统，一般很难用数学模型来描述，它的属性一般采用流程图来描述。

模型是系统的一种抽象描述，是通过反复对系统进行分析研究而得到的系统的内在联系及其与外界的关系的一种描述。它的理论基础是相似原理。在进行系统仿真时用到的模型主要是实体模型和数学模型。实体模型是指根据相似性建立起来的系统的物理模型。数学模型是指对系统进行抽象、简化，能够准确表达系统本质的由数学符号表示的一种模型形式。由于实体模型使用起来不经济而且耗时，数学模型具有形成简单、应用方便、经济，便于使用计算机技术等优点，所以在系统仿真中通过模型描述系统时一般都采用数学模型。

系统仿真技术是随着微电子分析器在 1946 年的诞生而迅速发展起来的新兴综合性学科。以相似原理、系统技术、信息技术及其应用领域相关专业知识和技术为基础，以计算机系统和各种相关的器件为工具，利用系统模型对实际存在的或是假想的系统进行动态研究的一门多学科的综合性技术。随着系统仿真技术理论方法和应用技术研究的不断深入，计算机技术的突飞猛进，以及系统仿真本身具有的安全性、经济性等特点，应用计算机对系统进行仿真在科学技术领域中发挥着越来越重要的作用。需要特别指出的是，系统仿真使用系统模型代替实际系统来进行系统性能的分析和研究，因此使仿真更加具有意义。近些年来，随着计算机技术、网络技术、信号处理、通信技术、自动控制技术等高新技术的迅猛发展，系统仿真技术的研究力度也在不断加大，发展速度不断加快，应用领域不断扩大，系统仿真技术的应用已经从早期的航空航天、武器制造和发电部门，扩展到今天的军事、通信、控制、机械、经济、社会、交通与生态研究等众多领域，而且已经渗透到系统的规划、设计、运行、分析及改造的各个阶段。

系统仿真技术作为一门独立的学科，已经具有 60 多年的发展历史。它的发展主要分为两个阶段，第一个阶段是计算机出现之前，主要利用物理和数学进行系统的建模、仿真与分析；第二个阶段是计算机系统仿真阶段。从计算机系统仿真技术的发展历程来看，它又经历了五个阶段：

20 世纪 40 年代的模拟计算机仿真；

20 世纪 50 年代的数值计算机仿真；

20 世纪 60 年代的仿真语言的出现；

20 世纪 80 年代的面向对象的仿真技术；

20 世纪 90 年代的虚拟现实仿真技术和可视化的建模与仿真。

随着计算机技术的迅速发展和广泛应用，近 20 年来，国内外出现了许多专门用于计算机仿真的语言及软件工具，如：CSMP、ACSL、SIMNON、MATLAB/SIMULINK、CSMP_C 等，而 MATLAB/SIMULINK 的出现，不仅使数值分析与应用进入了一个崭新的阶段，而且也为系统仿真技术提供了更实用、更方便的解决办法。本书就是以目前仿真领域最权威、最实用的计算机仿真工具——MATLAB/SIMULINK 为基础，介绍需要求解的问题的解决方法及其在系统仿真领域的应用。

1.2 MATLAB仿真技术的发展与应用

MATLAB(Matrix Laboratory,即“矩阵实验室”)最初由美国新墨西哥大学Cleve Moler教授提出并应用于矩阵计算的教学中。自从MATLAB由Mathworks公司推出后,MATLAB为各国的工程科学家开发学科应用软件提供了新的基础,并且得到了全世界的关注和欢迎。MATLAB经过不断地扩充和完善,已经发展成为功能强大,适用于多个学科和领域的系统软件工具。

Mathworks公司于1984年推出MATLAB商业版,1990年推出了第一个可以运行在Microsoft Windows系统下的MATLAB 3.5i版本,1992年推出了MATLAB 4.0版本,两年之后又推出了MATLAB 4.2版本。MATLAB 4.x版本除了保持和改善了原有的数值计算和图形功能以外,还新增了许多实用的新功能:

- 推出了SIMULINK动态系统的建模、仿真和分析的集成仿真环境。SIMULINK的出现提高了人们处理非线性因素和随机因素的能力。
- 增加了与外部直接进行数据交换的单元。
- 增加了符号运算软件包。
- 增加了Notebook,这使得MATLAB和Word文档能够更加容易地连接起来。

随后,在1997年又推出了MATLAB 5.0版本,1999年推出了MATLAB 5.3/SIMULINK 3.0版本。2000年推出了MATLAB 6.0/SIMULINK 4.0版本,一年之后,MATLAB 6.1/SIMULINK 4.1问世了。现在MATLAB /SIMULINK最高的版本是MATLAB 6.5/SIMULINK 5.0。随着版本的不断更新,MATLAB的功能不断完善,已不再仅仅是“矩阵实验室”,其功能齐备的应用工具箱、良好的用户界面、便捷的模块化动态仿真环境,使MATLAB/SIMULINK成为国际上最为流行的科学计算以及系统仿真领域的系统软件工具。

今天的MATLAB集数值计算、图形可视化、图像处理以及多媒体技术于一身,在航天、航空、军事、经济、交通、自动控制、通信、信号处理、系统优化设计等众多领域得到广泛的应用。现在,MATLAB已经成为国内外大学生和研究生进行学习以及科学研究时必须掌握的基本软件工具,在设计研究和工业部门,MATLAB也广泛应用于研究和解决各种实际的工程问题。

1.3 MATLAB仿真技术的特点

MATLAB经过长期不断地发展和完善,已经成为当今世界上最优秀的数值计算软件,受到了人们的广泛欢迎。

MATLAB除了具有强大的科学计算功能和丰富的图形功能之外,还具有一些其他软件无法比拟的功能和优点:

- 具有API应用程序接口,能够保证MATLAB便捷地和其他强大的计算机软件

相结合。

- 具有较高的计算精度，通常数值计算精度能够达到 10^{-15} 数量级，能够满足科学计算和工程的要求。
- 基本的运算单位是矩阵，操作数组像操作数一样方便，使用户易读易写。
- 具有源程序的开放性和功能齐备的软件工具包。MATLAB 丰富、实用的工具包，节省了用户编写复杂的应用程序所花费的时间，只需调用相应的工具箱中的函数就可以了。
- 具有世界一流水平的数学函数库。
- 完全的基于 HTML 的帮助系统。
- 语言简洁紧凑，程序设计流程灵活，易学易用，扩充能力强。

另外，MATLAB 还有一个显著的特点，就是提供了系统动态仿真工具箱——SIMULINK。SIMULINK 使得 MATLAB 的功能得到了进一步的扩展。SIMULINK 由模块库、模型构造及指令分析和演示程序组成，是一个模块化、模型化的系统动态仿真环境。用户应用 SIMULINK 对系统进行建模、仿真和分析时如同堆积积木一样简单方便，只需要在模型窗口中单击或是拖动鼠标即可。SIMULINK 不能脱离 MATLAB 而独立运行，但是它借助 MATLAB 在科学计算上得天独厚的优势以及可视化的仿真模型窗口，弥补了传统软件工具的不足。同时，SIMULINK 也是众多仿真软件中功能最强大、最优秀的一种软件工具。它使得动态系统仿真的实现相当方便，对系统的非线性因素和随机因素的研究也十分便捷、直观。通过 SIMULINK 还可以对电气、机械、通信等的连续、离散或是混合系统进行深入的系统建模、仿真与分析研究。

正是因为 MATLAB/SIMULINK 具有众多其他同类软件不具备的优点，所以才受到国内外学者和工程师的备加关注，得以不断地扩充和迅速发展，成为当今世界在科学计算和系统仿真领域里首选的软件工具。

1.4 仿真应用实例简介

本节将通过 MATLAB/SIMULINK 在系统仿真中的应用实例来介绍 MATLAB/SIMULINK 所具有的功能和特点。在这里，将不对相关语句或仿真模型模块的具体含义作详细解释，只是让读者对 MATLAB/SIMULINK 的功能与应用有初步的了解，其相应解释将在本书后续章节中详细介绍。

例 1－1 对一个简单的系统，在输入为阶跃响应情况下进行时域分析，并计算动态性能指标（超调量、上升时间和调节时间）和系统的零极点。系统的传递函数为：

$$G(s)=\frac{Y(s)}{X(s)}=\frac{5}{s^2+2s+10}$$

其仿真程序代码如下：

```
% this program performs an analysis of its step response
num=5;den=[1,2,10];
[z,p,k]=tf2zp(num,den)
```

```
[y,x,t]=step(num,den);
t1=length(t);yss=y(t1);
[ym,tm]=max(y);
singma=100*(ym-yss)/yss          %计算超调量
n=1;                             %计算上升时间
while y(n)<0.1*yss
    n=n+1;
end
m=1;
while y(m)<0.9*yss;
    m=m+1;
end
risetime=t(m)-t(n)
while y(t1)<1.02*yss & y(t1)>0.98*yss       %计算调节时间
    t1=t1-1;
end
stime=t(t1)
plot(t,y)
grid on
```


系统的阶跃响应曲线如图 1-1 所示。

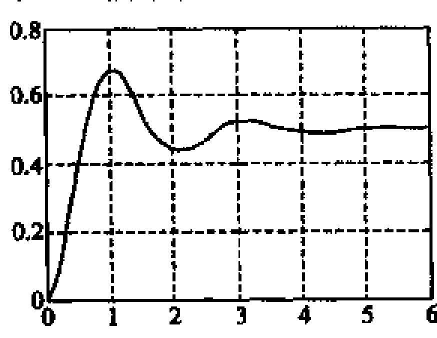


图 1-1 系统阶跃响应曲线

运行结果如下：

```
z =
  Empty matrix: 0-by-1
p =
    -1.0000 + 3.0000i
    -1.0000 - 3.0000i
k =
    5
singma =
    35.1958
risetime =
    4417
stime =
```

3.5337

这个实例是 MATLAB 在控制系统时域分析中的简单应用，从这里可以看出它具有强大的计算功能以及丰富的函数库。

例 1-2　当（$-3<x<3$，$-3<y<3$）时，运行下面的程序将产生函数 $x*(-x^2-y^2)$ 的三维图形。

运行下面的程序代码，将得到如图 1-2 所示的三维图形。

```
[X,Y] = meshgrid(-3:.1:3, -3:.1:3);
Z = X .* exp(-X.^2 - Y.^2);
mesh(Z)
```

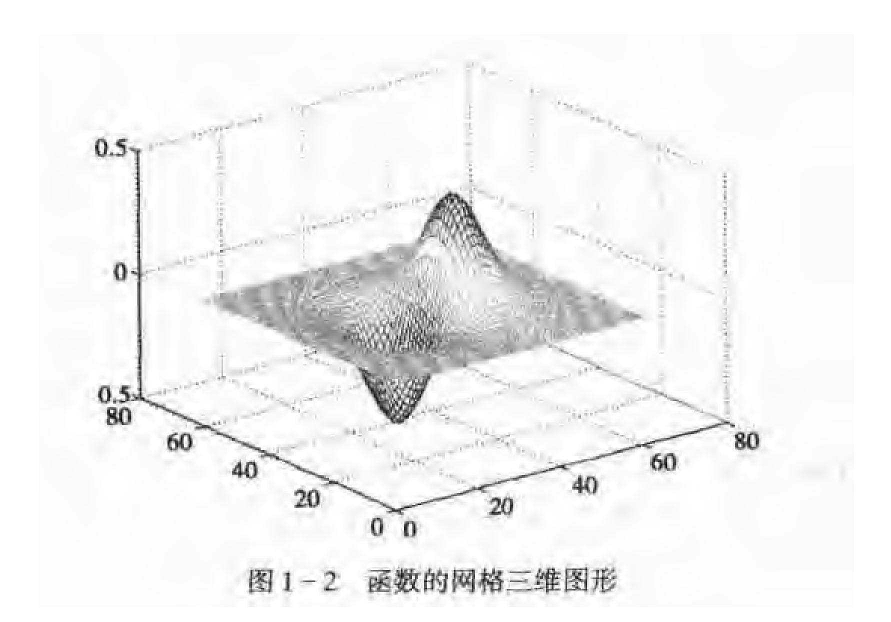


图 1-2　函数的网格三维图形

从这个实例可以看出，MATLAB 具有强大的图形绘制以及可视化功能，它拥有丰富的图形处理命令，能够绘制二维、三维图形，方便了用户对图形的加工处理。面绘制和处理复杂的图形、图像，更能够体现 MATLAB 所具有的强大优势。

例 1-3　这是一个计数器的演示程序。该演示程序是使用计数器电路，在相同的控制信号下，运行使能计数器和触发计数器子系统模块的情况。在 SIMULINK 仿真环境下，系统的模型如图 1-3 所示。使能计数器和触发计数器子系统模块模型如图 1-4 所示。

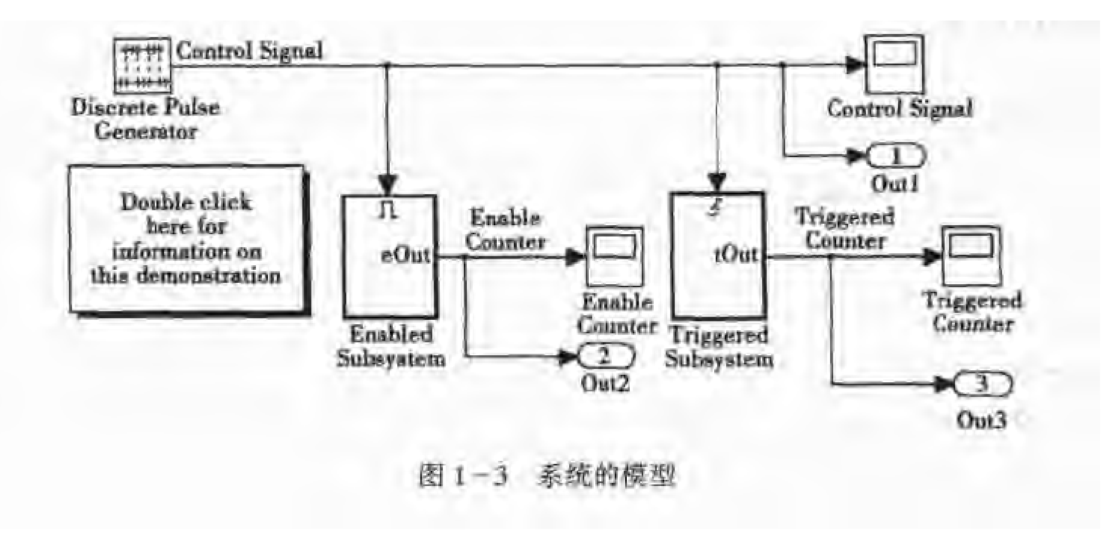


图 1-3　系统的模型

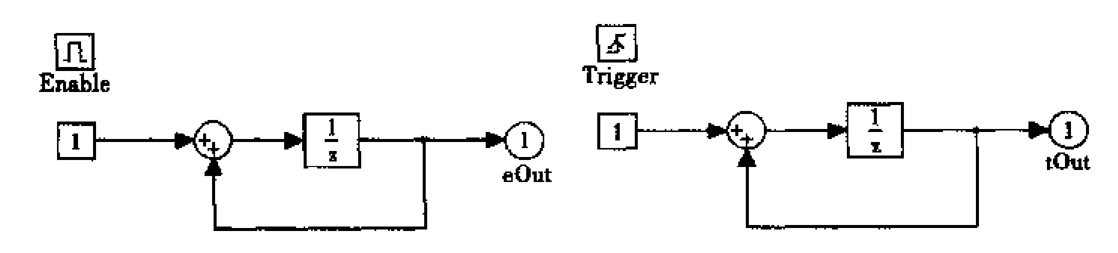


图1-4 使能子系统与触发子系统的模型

运行仿真，可以得到在相同的控制信号(如图1-5所示)作用下，使能计数器(enabled counter)与触发计数器(triggered counter)子系统模块的输出波形，如图1-6所示。

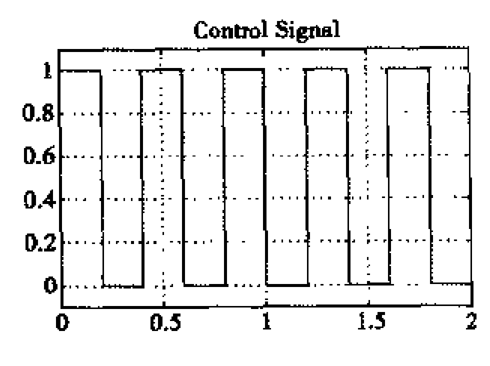


图1-5 系统控制信号

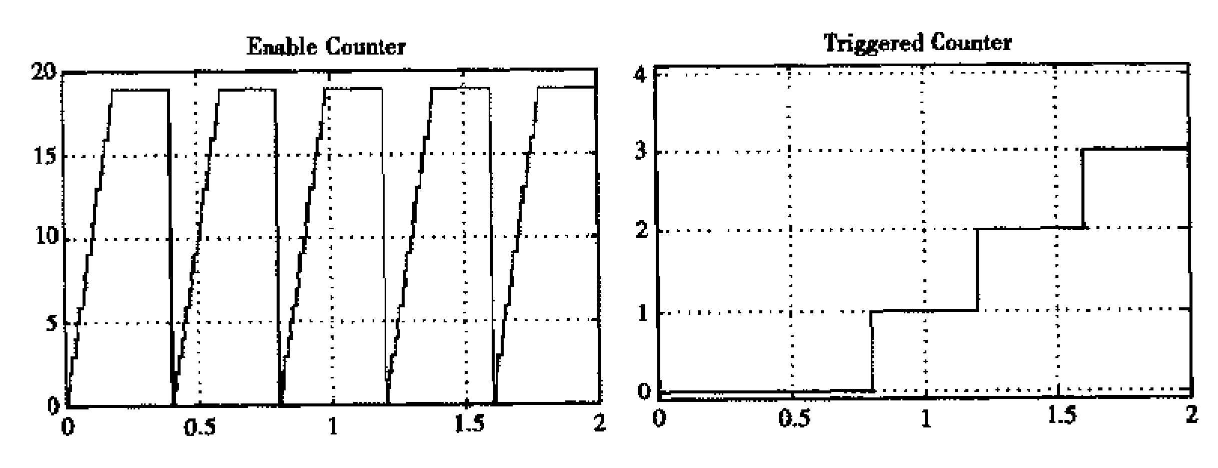


图1-6 使能计数器与触发计数器子系统模块输出波形

从以上实例可以看出利用SIMULINK进行系统动态仿真的便捷性与直观性。此外，MATLAB还提供了许多计算和应用实例，在MATLAB命令窗口中直接输入demo命令即可获得所有实例，这对于用户全面了解MATLAB/SIMULINK有很大的帮助。

1.5 MATLAB网络资源

当今，网络技术已经相当发达，网络已成为人们生活不可缺少的一部分，它为人们的生活、工作、学习带来了极大的方便。MATLAB的网络资源十分丰富，包括新闻组、FIP

资源以及各种 Web 网站等，从这里，用户可以直接获得各种资源，如：产品介绍、学术动态、技术文档、软件下载与升级、技术帮助等等。在互联网上获得各类信息的最简便、灵活的方法是从相关 Web 网站获得。本节将介绍如何在 Web 网站上直接获得有关 MATLAB 的各类资源。

Mathworks 公司建立了自己的网站，这里是 MATLAB 的网上老家，是 Mathworks 公司用来向它的网络用户提供各种信息的地方。其网址为：http://www.mathworks.com。从该公司的网站上可以查询各种信息，包括 Mathworks 公司以及其有关产品的介绍，用户组讨论区，在线复制通信季刊，最新的 MATLAB 会议论文集，全套的 MATLAB 联机帮助文件和说明书的电子版等等。另外，还可以获得各种技术支持与帮助。

由薛定宇教授致力维护的 MATLAB 语言及其在科学计算和控制系统中应用的网站，在国内是最受欢迎的 MATLAB 网站，其网址为：http://matlab.myrice.com。这个网站不仅有着大量的 MATLAB 相关信息以及免费软件下载，还有"MATLAB 语言及应用论坛"，在这个论坛中，每一位访问者都可以对 MATLAB/SIMULINK 编程和仿真的方法与问题发表自己的意见和看法，经常会有国内外本领域的专家就各类疑难问题进行解答。

http://mathtools.top263.net 网站被称为 MATLAB/SIMULINK 的资料库，它提供了大量的资料供 MATLAB 用户免费下载。它的特色是侧重 MATLAB 语言与 C 语言接口的讨论。

此外，还有一些办得较好的有关 MATLAB 的网站，在那里不仅可以获得最新的 MATLAB 信息和软件下载，同时，还可以进入讨论区与其他用户和专家进行讨论、交流。这类网站主要包括：

- http://www.mathtools.com
- http://passmatlab.myetang.com
- http://europe.mathworks.com
- http://www.geatbx.com
- http://huxuan.home.china.com
- http://comp.soft_sys.matlab
- http://www.mathtools.net
- http://www.indiana.edu
- http://www.math.mtu.edu
- http://matlab.turbo.hit.edu.cn

1.6 本章小结

本章简要介绍了 MATLAB 系统仿真技术及应用，目的是让读者在学习后续章节前对 MATLAB/SIMULINK 有一个整体的理解和感性的认识。

系统是仿真的研究对象，模型是仿真的基础，了解了模型与系统的基本概念之后才能全面而准确地理解系统仿真技术。MATLAB 仿真技术的发展与控制工程、系统工程及计算机的发展有着密切的联系。它经过多年的发展与扩充，已经从开始的"矩阵实验室"发

展成为现在世界上应用最广泛、最受人们欢迎的进行系统仿真与科学计算的软件工具，并且在仿真领域的软件市场占有主导地位。MATLAB 仿真技术是在国内外各个领域的专家的参与下不断完善和流行起来的。它具有强大的计算和绘图功能、源程序的开放性、功能齐备的应用工具箱、语言简单、易读易写等优点。

另外，通过国内外比较知名的 MATLAB 的网站，用户可以直接获得大量的 MATLAB 相关信息与资料。

习　题

1. 什么是系统、模型以及系统仿真技术?
2. 系统仿真技术的发展历程分为几个阶段?
3. MATLAB/SIMULINK 仿真技术的特点有哪些?
4. 试从 MATLAB 所提供的演示程序中了解 MATLAB 仿真技术的应用。
5. 本章中提供了许多有关 MATLAB 资料的网站，试从中查询并了解一些与自己专业相关的 MATLAB 资料。

第2章 SIMULINK动态仿真

知识点：

- 仿真模型的创建
- 动态仿真
- 使用命令进行仿真
- 仿真结果分析
- 子系统
- 动态仿真实例

本章主要介绍如何利用SIMULINK进行动态系统建模、仿真以及结果分析，并在此基础上深入地介绍建立仿真模型时简化模型并增加其可读性的方法，即创建子系统模块。同时，将通过仿真实例介绍实现系统动态仿真的过程与方法。

通过对本章的学习，读者可以掌握进行系统建模、仿真与分析的方法和技巧，子系统的创建、封装及使用方法。同时，还可以了解条件执行子系统及其应用，以及创建自己的模块库的方法。

2.1 仿真模型的创建

创建一个系统仿真模型就像搭积木一样，打开创建模型窗口，首先从相应的模块库中选出所需要的仿真模块，用鼠标把它们依次拖到模型窗口。然后用信号线把各个模块按照系统要求连接起来，组成所需要的系统仿真模型。本节主要介绍仿真模块、仿真信号线以及如何创建仿真模型。

2.1.1 仿真模块

仿真模块是创建SIMULINK仿真模型的基本单元，是SIMULINK进行动态系统仿真的基础。用适当的方法创建所需要的仿真模块并将其相互连接起来，就构成了一个系统模型。有时，为了构造系统模型需要对仿真模块进行一系列的操作。

1. 创建一个仿真模块

打开模块库之后，可以通过以下两种方法之一来创建一个仿真模块：

- 用鼠标左键选中所需要的模块，然后将其拖到需要创建仿真模型的窗口，松开鼠标，这时所需要的模块将出现在仿真模型窗口中。

- 利用鼠标右键来完成仿真模块的创建：首先选中所需要的模块，然后单击鼠标右键，在弹出的快捷菜单中选择 Add to file _ name 命令（其中 file _ name 是仿真模型的文件名），这样，仿真模块也将出现在 file _ name 窗口中。

2. 模块的复制操作

如果需要完成对仿真模块的复制操作，可以通过以下方法之一来实现：

- 选中所需复制的仿真模块，在按下 Ctrl 键的同时按住鼠标左键将模块拖到目标位置后释放鼠标。
- 选中所需复制的仿真模块，在菜单栏中执行 Edit/Copy 命令。
- 选中所需复制的仿真模块，首先单击鼠标右键，在弹出的快捷菜单中选择 Copy 命令，然后在目标位置单击鼠标右键，在弹出的快捷菜单中选择 Paste 命令。

3. 模块的移动操作

在仿真模型窗口中，如果想将一个仿真模块从原来的位置移动到一个新的位置，只需要选中模块，然后用鼠标左键将模块拖到目标位置即可，而与之相连接的信号线，则由 SIMULINK 自动重新绘制。如果需要实现对多个模块及与其相连接的信号线的同时移动，首先选定所需移动的所有模块和信号线，然后将其移动到目标位置即可。

4. 调整模块大小的操作

为了满足特殊需要（如使模块的外形更加美观，增强模型的可读性），有时候需要调整模块的大小。要完成对模块大小的调整操作，首先选中相应的模块，这时模块的四个对角点都将出现黑色的控制柄，单击任一控制柄，此时鼠标的指针形状将发生改变。然后按住鼠标左键向内或外拖动，即可实现对模块大小的调整。

注意：调整模块大小的操作，只是改变模块的外观，不会改变模块的各项参数。

5. 模块的删除操作

如果需要完成对仿真模块的删除操作，可以通过以下方法之一来实现：

- 选中需要删除的模块，在菜单栏中执行 Edit/Clear 命令。
- 选中需要删除的模块，在菜单栏中执行 Edit/Cut 命令，这样在删除模块的同时会将其保存在剪切板上。
- 选中需要删除的模块，按下键盘上的 Delete 键。
- 选中需要删除的模块，单击鼠标右键，在弹出的快捷菜单中选择 Cut 或 Clear 命令。这两个命令不同之处在于当选择 Clear 命令时将彻底删除模块，而选择 Cut 命令时，在将模块删除的同时会将其保存到剪切板上。

6. 模块的旋转操作

仿真模块的默认信号是从左端到右端，即左端是输入端，右端是输出端。有时为了模块间连线方便，需要对模块进行旋转操作。要对模块进行旋转操作，可以先选中待旋转的

模块,然后在菜单栏中执行 Format/Rotate 命令,这时模块将按顺时针方向旋转 90°;若在菜单栏中执行 Format/Flip 命令,模块则旋转 180°。如同模块的其他操作一样,单击鼠标右键,然后从弹出的快捷菜单中选择相应的命令,也可以完成对模块的旋转操作,具体方法这里不再赘述。

7. 模块参数的设置

在 SIMULINK 环境下绘制模块,只能给出带有默认参数的模块模型,为了满足用户的具体需要,有时还需要对模块参数进行具体的设置。要对模块进行参数设置,首先双击相应的模块,这时将打开此模块的参数设置对话框。在该参数设置对话框中,既可以查看模块的各项默认参数设置,也可以根据需要修改各项参数设置。图 2-1 为积分器的参数设置对话框。



图 2-1　模块参数设置对话框

另外,还可以通过 set _ param 仿真命令来完成对模块参数的设置。

8. 对模块标签的操作

SIMULINK 除提供了针对仿真模块的操作外,还提供了针对仿真模块标签的操作。下面简单介绍有关仿真模块标签的基本操作。

(1)*更改模块标签名*

一个仿真模块创建后,SIMULINK 会在模块下面自动生成一个以该模块命名的标签。如果用户需要更改模块标签名,可以双击该模块标签,使其处于编辑状态,然后输入新的标签名称即可。

(2)*改变模块标签位置*

如果需要改变模块标签的位置,可以首先选中模块,然后在菜单栏中执行 Format/Flip name 命令,此时标签的位置将发生旋转。此外,还可以通过拖动鼠标来完成此操作,

首先选中标签，然后用鼠标左键将其拖到新的位置即可。

注意：这里的新位置不是任意位置，若标签在模块下方，则标签只能被拖到模块上方；若标签在模块左侧，则标签只能被拖到模块右侧。

(3)隐藏模块标签

有时为了简化模型或是方便连线，可以将一些功能显而易见的模块的标签设置为隐藏。方法是：选中需要隐藏标签的模块，在菜单栏中执行 Format/Hide name 命令，这时即可完成对标签的隐藏。标签虽然被隐藏，但是它仍然存在，如果需要还可以将被隐藏的标签显示出来。其方法是选中需要显示标签的模块，在菜单栏中执行 Format/Show name 命令(当标签被隐藏后，Format 菜单中的 Hide name 命令将变为 Show name 命令)。以上操作同样可以通过单击鼠标右键在弹出的快捷菜单中完成。

9. 模块增加阴影操作

有时为了让模块引起用户的注意，可以为模块增加阴影。方法是：首先选中需要增加阴影的模块，然后在菜单栏中执行 Format/Show drop shadow 命令，即可完成相应的操作。以上操作同样可以通过单击鼠标右键在弹出的快捷菜单中完成。如果想除去模块的阴影，可先选定需要去掉阴影的模块，然后在菜单栏中执行 Format/Hide drop shadow 命令即可。

此外，在 SIMULINK 模型窗口的 Format 菜单中还提供了另外一些对模块进行修饰的命令。

2.1.2　仿真信号线

在 SIMULINK 系统仿真中，模块间的信号传输是通过连线来完成的，所以把模块间的连接线称为信号线。信号线上传输的信号可以是标量信号也可以是矢量信号。下面介绍信号线的连接以及针对它的一些操作。

1. 模块之间信号线的连接

要实现模块间信号线的连接十分简单和方便，具体的操作步骤如下：

(1)将鼠标移至源模块的输出端口，此时鼠标指针形状将变为十字形。

(2)拖动鼠标至目标模块的输入端口。

(3)松开鼠标即可完成模块连接操作。SIMULINK 用一条带有箭头的连线将模块连接起来，箭头方向表示信号的传输方向。

除了以上方法外，还可以通过首先选中源模块，然后在按下 Ctrl 键的同时，用鼠标左键单击目标模块的方法完成模块之间信号线的连接操作。通过以上两种方法画出的信号线是垂直的或者是水平的。

2. 绘制支路信号线

支路信号线是从已经存在的信号线上引出，将信号传输到另一个模块的输入端的信

号线。使用支路信号线可以将同一个信号传输给多个模块，要增加一条支路信号线可以采用以下方法之一：

- 按住 Ctrl 键，在需要建立支路信号线的地方按下鼠标左键并拖动鼠标至目标模块的输入端，然后松开 Ctrl 键和鼠标。
- 在需要建立支路信号线的地方按下鼠标右键并拖动鼠标至目标模块的输入端，然后松开鼠标。
- 用鼠标左键从目标模块的输入端拖出信号线到支路信号线的引出点，但要注意信号线一定要连接起来才能松开鼠标。

图 2－2 为将 Sine Wave 模块的信号同时输出给 Scope 模块和 Display 模块。

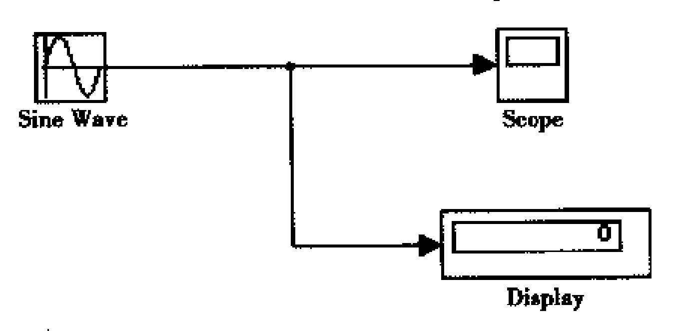


图 2－2　绘制支路信号线示例

3. 在信号线间插入模块操作

有时根据仿真模型的要求，需要在已经存在的信号线上插入一个新的模块，其具体操作步骤如下：

(1)选中需要插入的模块。

(2)拖动模块到信号线上需要插入模块的位置，松开鼠标即可。

注意：插入的模块只能有一个输入端和一个输出端。

图 2－3 为在 Sine Wave 模块和 Scope 模块间插入 Integrator 模块的操作过程。

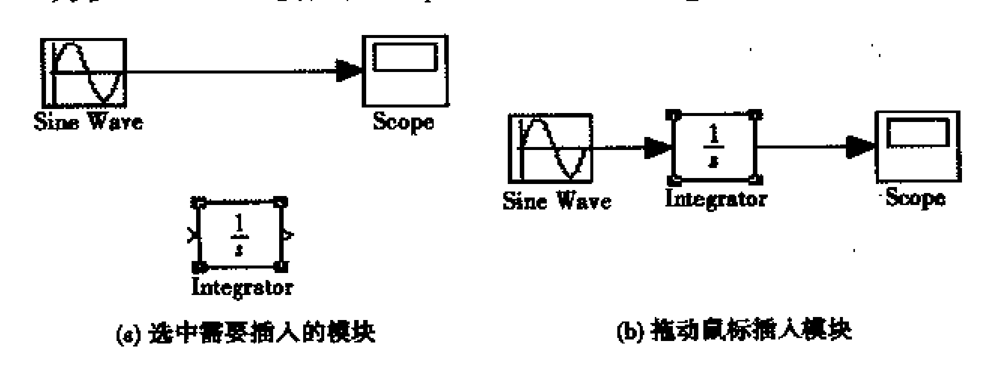


图 2－3　在信号线上插入模块操作过程

4. 分割信号线操作

在 SIMULINK 仿真环境下，还可以对信号线进行分割操作(即把直线变成折线的操作)。实现的具体操作步骤如下：

(1)选中需要分割的信号线。

(2)把鼠标移至分割点。

(3)同时按下 Shift 键和鼠标左键,此时会在信号线上出现一个小圆圈即分割点。

(4)用鼠标拖动分割点至所要求的位置。

(5)松开鼠标和 Shift 键即可。

图 2-4 为分割 Sine Wave 模块与 Scope 模块间的信号线的示例。

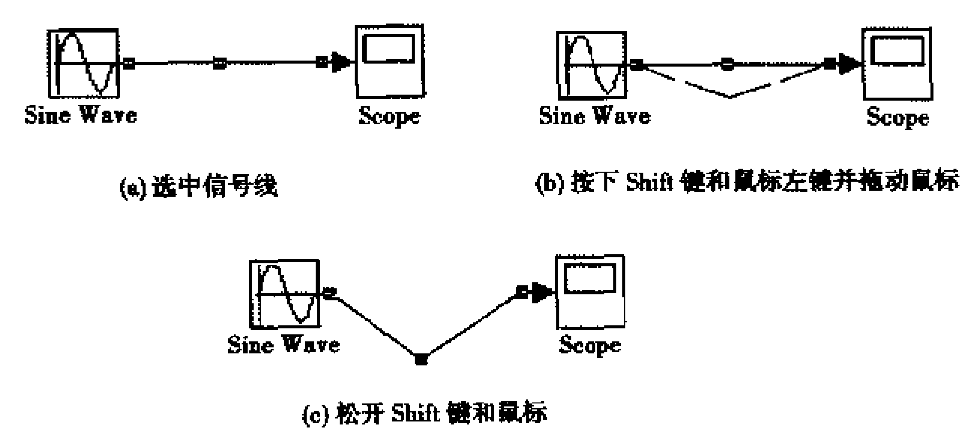


图 2-4　分割信号线的实例

5. 移动信号线操作

在信号线已经连接好的情况下,为了改善模型的外观和可读性,通常会对信号线进行移动操作。实现此操作的步骤如下:

(1)选中要移动的信号线。

(2)把鼠标移到信号线上,此时鼠标指针形状将变为四个方向都有箭头的十字形。

(3)拖动鼠标把信号线移动到所需要的新位置,松开鼠标。

6. 设置信号线标签操作

设置信号线标签的操作方法很简单,只需在需要输入标签的信号线上双击鼠标,当出现编辑标签的编辑框后就可以输入信号线的标签了,然后再用鼠标把信号线的标签拖到合适的位置即可。图 2-5 为在 Sine Wave 模块与 Scope 模块间的信号线上输入信号线的标签 L1 的操作过程。

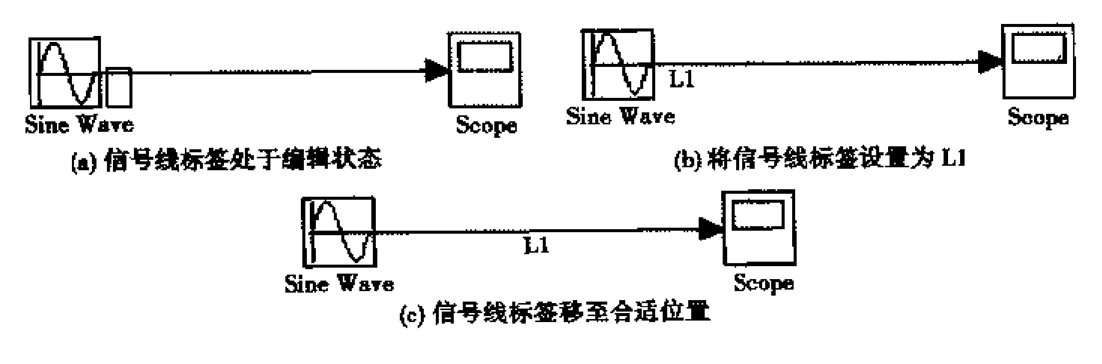


图 2-5　设置信号线标签示例

2.1.3 模型创建实例

通过对本章前面内容的学习，相信读者对仿真模块及其连接已经有了初步的了解和认识。接下来将通过实例具体介绍创建仿真模型的操作过程。

例 2－1 已知一个系统的微分方程表示为：

$$\begin{cases} \dfrac{dx_1}{dt} = x_2 \\ \dfrac{dx_2}{dt} = u - 5 \times x_1 \end{cases}$$

其中，状态变量初始条件 $x_1(0)=0, x_2(0)=0$，输入 u 为阶跃函数，要求利用 SIMULINK 对系统建立仿真模型同时将其以文件名“li2_1”保存，并绘制时域响应曲线。

在利用 SIMULINK 创建模型之前，先把微分方程进行拉普拉斯变换，得到每个微分方程的传递函数，即用传递函数的形式表示系统。得到传递函数后，首先打开 SIMULINK 模块库，把所需要的模块通过复制等操作放到创建模型窗口。这个系统所需要的模块有两个积分模块（Integrator 和 Integrator1）、一个增益模块（Gain）、一个求和器模块（Sum）、一个阶跃函数输入模块（Step）以及一个示波器模块（Scope），把这些模块拖放到模型窗口，如图 2－6 所示。

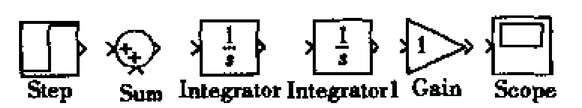


图 2－6 系统模型所需要的模块

然后根据系统的微分方程或传递函数，对各个模块进行参数设置。其中，将增益模块的放大倍数（Gain）设置为 5，求和器模块输入端的符号（List of signs）设置为一个为负，一个为正。这两个模块的参数设置对话框分别如图 2－7 和图 2－8 所示。完成各项参数设置后单击 OK 按钮即可。

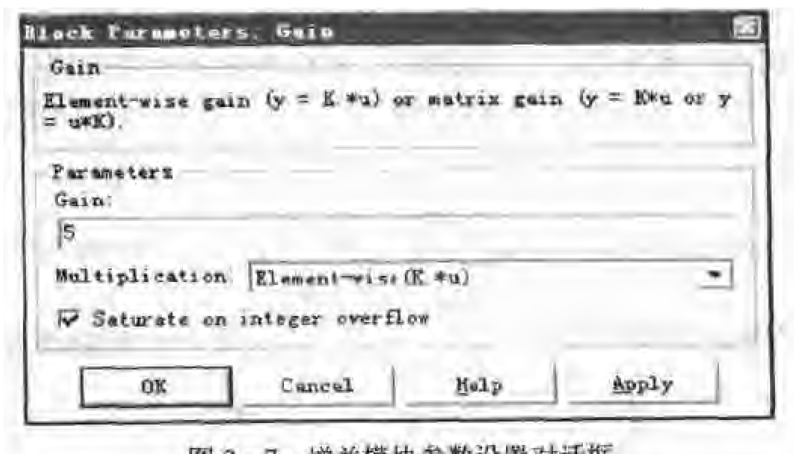


图 2－7 增益模块参数设置对话框

仿真模块参数设置完成之后，就需要连接各个模块之间的信号线了。为了连接信号线方便，可以先对增益模块进行旋转操作，然后再进行信号线的连接。信号线连接完毕的

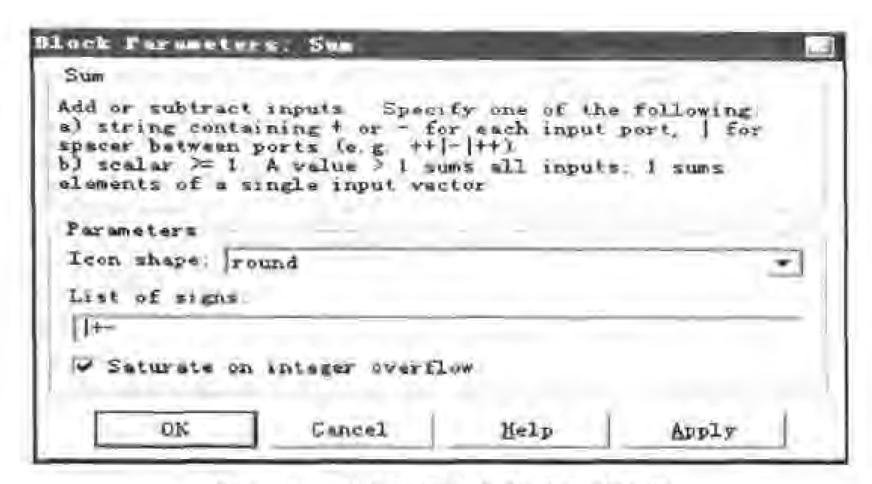


图 2－8　求和器模块参数设置对话框

系统模型如图 2－9 所示。设置信号线的标签属性，如图 2－10 所示。

系统模型建立之后就可以进行系统仿真。在进行系统仿真之前，应先把 Step 模块的 Step time 参数设置为从零开始，然后在仿真模型窗口的菜单栏中执行 Simulation/Start 命令，即可以开始系统仿真。仿真结束后双击 Scope 模块，即可以看到系统在阶跃输入情况下的时域响应曲线，如图 2－11 所示。

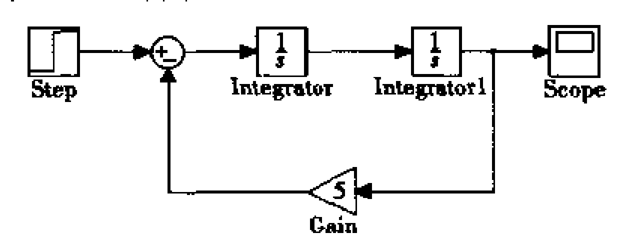


图 2－9　连接信号线之后的系统模型

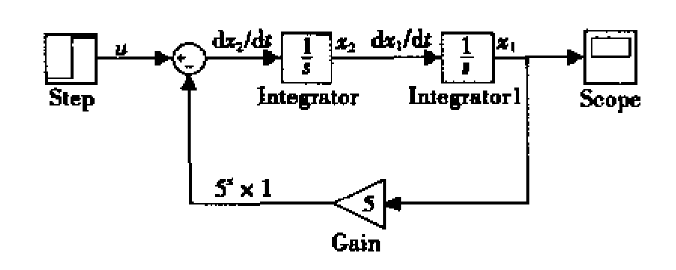


图 2－10　系统仿真模型

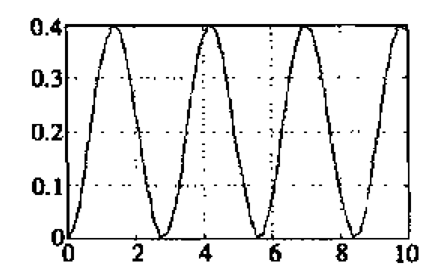


图 2－11　系统时域响应曲线

2.2　SIMULINK 动态仿真

在 2.1 节中介绍了仿真模块和信号线以及创建系统仿真模型的方法。本节将介绍仿真时如何使用 SIMULINK 调试器以及如何在菜单窗口运行系统仿真。

2.2.1　SIMULINK 调试器

无论是 SIMULINK 的初学者,还是 SIMULINK 的高手,建立的系统仿真模型都不能保证完全正确,SIMULINK 调试器就是用来帮助用户寻找和诊断模型中可能存在的错误的工具。它允许用户单步执行仿真,可以显示模型的即时状态和模块的输入输出,以便查明模型错误。SIMULINK 的调试方法有两种,即图形界面调试器和命令调试。接下来介绍如何使用这两种调试方法来诊断 SIMULINK 仿真模型中可能出现的问题和存在的错误。

1. 使用命令进行调试

SIMULINK 在 4.0 版本之前,使用的调试方法都是基于命令行输入的,虽然图形界面调试器简化了调试操作,对于初学者也比较简单,但是使用命令行进行调试却更加灵活、快捷。

在命令窗口中启动调试器可以通过 Sim 命令或 Sldebug 命令实现。例如以 2.1 节中图 2－9 所示的仿真模型为例,可在命令提示符下输入如下命令:

```
>>[t,x,y]=sim('li2_1',[0,10],simset('debug','on'));
```

或

```
>> sldebug 'li2_1';
```

这两个命令都将把 SIMULINK 仿真模型'li2_1'加载到内存,同时,在第一个仿真时间步内暂停在执行顺序中的第一个模块,并且高亮显示模型的起始模块以及与之相连接的输出信号线,如图 2－12 所示。

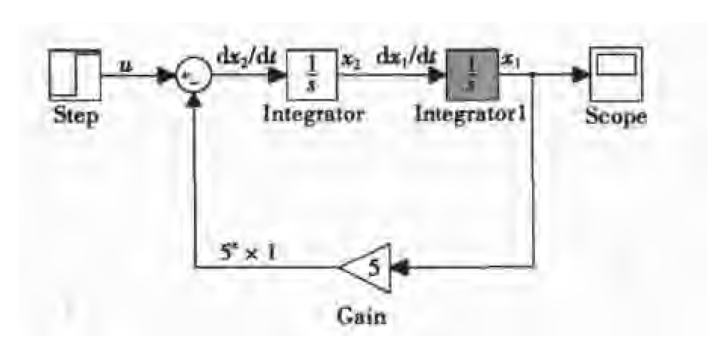


图 2－12　用命令启动调试器

使用命令进行调试,调试器将在 MATLAB 命令窗口中显示仿真的开始时间和调试命令的提示符。命令提示符下将显示模块的索引和将要被执行的第一个模块的名称。如执行 Sim 命令和 Sldebug 命令后都会显示如下内容:

```
Warning: Temporarily disabling 'Signal storage reuse' to allow use with sldebug.
Warning: Simulink debugger is disabling block reduction option.
[Tm = 0                    ] **Start** of system 'li2_1' outputs
%------------------------------------------%
(sldebug @0:0 'li2_1/Integrator1'):
```

此时,用户可以在调试命令提示符后,输入调试命令或者其他的 MATLAB 命令,如

单步执行、显示指定模块或显示仿真信息等。另外，用户还可以输入 help 命令来获取调试器命令的帮助信息。表 2－1 为部分调试器命令的帮助信息，读者在使用命令进行调试时可以借鉴参考。

表 2－1　调试命令参考信息

<table>
<tr><th>命　令</th><th>是否可以重复</th><th>功能说明</th></tr>
<tr><td>ASHOW<GCB|S:B|S#N|CLEAR></td><td>否</td><td>显示代数环</td></tr>
<tr><td>ATRACE LEVEL</td><td>否</td><td>设置代数环的跟踪等级</td></tr>
<tr><td>BAFTER GCB|S:B</td><td>否</td><td>插入在模块执行后生效的断点</td></tr>
<tr><td>BREAK GCB|S:B</td><td>否</td><td>插入在模块执行前生效的断点</td></tr>
<tr><td>BSHOW S:B</td><td>否</td><td>显示指定模块</td></tr>
<tr><td>CLEAR GCB|S:B</td><td>否</td><td>清除模块断点</td></tr>
<tr><td>CONTINUE</td><td>是</td><td>继续执行仿真</td></tr>
<tr><td>DISP GCB|S:B</td><td>是</td><td>仿真停止时显示模块在停止时刻的输入输出</td></tr>
<tr><td>HELP</td><td>否</td><td>显示调试器命令的帮助信息</td></tr>
<tr><td>ISBOW</td><td>否</td><td>启用或禁止显示积分信息</td></tr>
<tr><td>MINOR</td><td>否</td><td>启用或禁止最小步长模式</td></tr>
<tr><td>NANBREAK</td><td>否</td><td>设置或清除无穷大值断点</td></tr>
<tr><td>NEXT</td><td>是</td><td>单步执行到下一个时间步</td></tr>
<tr><td>PROBE[<GCB|S:B>]</td><td>否</td><td>显示模块的输入和输出</td></tr>
<tr><td>QUIT</td><td>否</td><td>退出仿真</td></tr>
<tr><td>RUN</td><td>否</td><td>运行仿真至结束</td></tr>
<tr><td>SLIST</td><td>否</td><td>列举非虚拟模块和模块的执行顺序表</td></tr>
<tr><td>STATES</td><td>否</td><td>显示当前的状态值</td></tr>
<tr><td>SYSTEMS</td><td>否</td><td>列举非虚拟系统</td></tr>
<tr><td>STATUS</td><td>否</td><td>显示当前调试选项</td></tr>
<tr><td>STEP</td><td>是</td><td>单步执行到下一个模块</td></tr>
<tr><td>STOP</td><td>否</td><td>终止仿真</td></tr>
<tr><td>TBREAK[T]</td><td>否</td><td>设置或清除时间断点</td></tr>
<tr><td>TRACE GCB|S:B</td><td>是</td><td>模块每次执行时显示模块的输入输出</td></tr>
<tr><td>UNDISP GCB|S:B</td><td>是</td><td>将模块从调试器的 DISPLAY POINTS 列表中除去</td></tr>
<tr><td>UNTRACE GCB|S:B</td><td>是</td><td>将模块从调试器的 TRACE 列表中除去</td></tr>
<tr><td>XBREAK</td><td>否</td><td>调试器遇到限步长状态时中断</td></tr>
<tr><td>ZCBREAK</td><td>否</td><td>设置非采样过零点事件断点</td></tr>
<tr><td>ZCLIST</td><td>否</td><td>列出可能包括非采样过零点的模块</td></tr>
</table>

对表 2－1 中涉及到的各个参量说明如下：

- 参量 S:B：索引为 S 的系统内的索引为 B 的模块。
- 参量 GCB：当前模块。
- 参量 S#N：系统 S 的标号为 N 的代数环。
- 参量 CLEAR：清除代数环高亮显示的开关。
- 参量 LEVEL：跟踪等级(0＝none,4＝everything)。
- 参量 T：需要设置断点的时间值。

2. 图形界面调试器

使用 SIMULINK 图形界面调试器相对于使用命令调试来查找仿真模型的错误较简单和方便。在模型窗口的菜单栏中执行 Tools/Simulink debugger 命令，则会启动 SIMULINK 图形界面调试器。“li2_1”图形界面调试器窗口如图 2－13 所示。



图 2－13　图形界面调试器窗口

从图 2－13 可以看出，整个调试器窗口分为三个部分即工具栏、Break on conditions（调试控制选项）、信息显示选项卡（Outputs、Execution Order 和 Status）。

调试器窗口的三个信息显示选项卡用来显示模型调试时的信息，控制选项和工具栏快捷按钮则提供了一些调试操作的图形命令。下面对这三部分分别作详细介绍。

1）工具栏

工具栏中的每一个快捷按钮都代表一个常用的调试命令。它们的具体功能如下：

- Step to next block 按钮（ ）：单击该按钮将运行仿真到下一个模块。
- Go to start of next time step 按钮（ ）：单击该按钮将运行仿真到下一个时间步。
- Start/Continue 按钮（▶）：该按钮用于开始或继续运行仿真。
- Stop debugging 按钮（■）：该按钮用于终止对模型的调试。
- Break before selected block 按钮（ ）：单击该按钮将在当前选择的模块运行前设置断点。
- Display I/O of selected block when executed 按钮（ ）：单击该按钮将在当前选择的模块运行前设置显示点，当选择的模块被执行时将显示它的输入和输出。
- Display current I/O of selected block 按钮（ ）：单击该按钮将显示当前选择的模块当前时间步的输入和输出。
- Help 按钮（?）：单击该按钮将显示调试帮助信息。

2）调试控制选项

图形界面调试器窗口的左边提供了调试控制选项，供用户进行条件断点的设置，它包

括四个复选框和一个文本框，具体说明如下：

- Zero crossings 复选框：当选中该复选框时，表示遇到过零点检测时产生断点。
- Step size limited by 复选框：当选中该复选框时，表示在步长受到限制的情况下产生断点。
- Minor time steps 复选框：当选中该复选框时，表示使仿真进入最小时间步长模式。
- NaN values 复选框：当选中该复选框时，表示在仿真时遇到无限大或超出机器表示范围的数值时产生断点。
- Break at time 文本框：该文本框用于输入时间值，表示在某个时间步设置断点。用户在文本框中需要输入设置断点的实际时间，而不是仿真步数。

3)信息显示选项卡

图形界面调试器窗口的右边区域用来显示仿真模型信息，它包括 Outputs、Execution Order 和 Status 三个选项卡，单击相应的标签将打开与之对应的选项卡。在启动调试器窗口时，只是在 Outputs 选项卡中显示相应的注释信息，其他两个选项卡将不显示任何信息。当开始进行调试时，三个选项卡将分别显示各自相应的仿真信息，打开相应的选项卡即可查看。

Outputs 选项卡用于显示调试执行后的输出结果，它显示的内容与在 MATLAB 命令窗口中显示的内容是一样的。Execution Order 选项卡用于显示当前模型中各个模块的排序表，同时给出各个非虚拟模块的索引信息，这个选项卡中显示的信息在整个调试过程中不发生任何变化。Status 选项卡用于显示调试器当前的设置信息，它的作用和在命令行输入 Status 命令是一样的，它所显示的内容在调试的过程中会随着用户的操作而变化，例如用户设置新的断点操作就会使此选项卡中的显示信息发生变化。

注意： 无论是设置断点，还是设置显示点，操作时都必须返回到仿真模型窗口，先选中相应的模块，再对其进行相应的操作。此外，对于设置了断点或是显示点的模块，SIMULINK 会自动把模块名称添加到 Break/Display points 列表框中。

2.2.2 运行仿真

建立好了系统的仿真模型之后，下一步就可以运行该模型进行仿真了。在本小节中将介绍如何使用菜单命令运行仿真，即使用窗口运行仿真。使用这种方法运行仿真十分方便，用户不必记住繁琐的 MATLAB 语法和指令就可以轻松地进行修改各项仿真参数和选择常微分方程解法等操作，它的最大优点是用户和模型之间交互性非常强，可以在仿真过程中随时进行相应操作。其中主要包括：

- 可以修改仿真参数，如仿真的求解解法、步长大小和时间范围等。
- 可以同时进行多个系统的仿真。
- 可以查看信号线上传递的信号取值。这个操作通过单击模型间信号线查看浮动的 Scope 或 Display 来实现。

- 可以修改模块参数。在修改模型参数时,有些参数值是不能修改的。如模块状态及输入输出个数、采样时间、过零点的个数、模块参数的矢量长度、内部模块工作的矢量长度等。

在仿真运行过程中,SIMULINK不允许用户对仿真模型的结构进行修改,例如增加或删除模块和信号线等操作。若要进行修改,则必须停止仿真,修改操作完成之后再进行仿真,这时将会得到模型修改后的仿真结果。

当需要运行仿真时,可以在SIMULINK仿真模型的仿真窗口菜单栏中执行Simulation/Start命令,或者在工具栏中单击▶按钮。要完整地完成一个模型的仿真,一般需要设置仿真参数、开始运行仿真和查看仿真结果三个步骤。

1. 设置仿真参数

在模型窗口菜单栏中执行Simulation/Simulation parameters命令,将打开如图2-14所示的仿真参数设置对话框。在该对话框中可以完成对各项仿真参数的设置。从图2-14可以看出,仿真参数设置对话框包括Solver(仿真解法设置)、Workspace I/O(工作空间输入输出)、Diagnostics(诊断)、Advanced(高级设置)和Real-Time Workshop(实时工具设置)等五个选项卡,单击相应的标签将会打开与之对应的选项卡。接下来,介绍其中主要的选项卡。

1)Solver选项卡

当单击Solver标签时,将打开Solver选项卡,如图2-14所示。Solver选项卡可分为Simulation time、Solver options、Output options等三个选项区域。接下来,分别对这三个选项区域的作用加以介绍。



图2-14 仿真参数设置对话框

◆ **Simulation time(仿真时间)选项区域**

在Simulation time选项区域中包括Start time和Stop time两个文本框。通过修改Start time和Stop time文本框中的数值可以更改仿真的开始和结束时间,它们的默认值分别是0.0s和10.0s。

注意：仿真时间和实际仿真用时间不相同，这与模型的结构复杂性及积分解法的步长等因素有关。

◆ **Solver options（仿真解法设置）选项区域**

对 SIMULINK 模型进行动态系统仿真时，通常都会涉及微分方程的求解问题，为此，SIMULINK 提供了多种积分求解方法以供用户使用。而动态系统又是复杂和不确定的，因此，为了准确、快速地解决不同的问题，就要求用户针对不同的问题选择不同的解法以及设置相应的参数。

- Type（解法类型）。仿真解法可以分为两大类：固定步长解法和变步长解法。固定步长解法是指在仿真过程中不改变积分步长的大小，这类解法不提供误差控制和过零点检测。变步长解法是指在仿真过程中根据条件的改变而改变积分步长大小，这类解法提供误差控制和过零点检测。

从 Solver 选项卡中可以看到，如果用户不对微分方程的解法进行选择，SIMULINK 提供有默认的解法。默认解法的选择是 SIMULINK 通过判断当前模型中是否有连续状态量来实现的。若仿真模型中有连续状态量，SIMULINK 则选择 ode45 解法，对于大多数情况下，选择 ode45 解法是非常有效的，但是当模型中存在刚性方程问题并且 ode45 解法不能求出仿真真实解时，可以采用 ode15s 解法。若仿真模型中没有连续状态量，SIMULINK 则选择变步长离散解法，同时给出没有使用 ode45 解法的信息。

- 步长设置。步长设置中包括 Max step size（最大步长）、Min step size（最小步长）和 Initial step size（初始步长）。对于固定步长解法，只能设置固定步长；对于变步长的解法，可以设置最大步长和最小步长。

SIMULINK 给出的默认值是 auto，最大步长用来限定最大时间间隔，默认值由起始和结束时间确定。计算公式为：

$$h_{max}=\frac{t_{stop}-t_{start}}{50}$$

通常系统默认的最大步长是足够用的，但是当仿真过程太长时，由上式定义的步长将会很长，这将导致错过一些具体的仿真过程，这时就要求用户应根据实际情况来确定最大步长的值。最小步长可以是一个大于零的实数值，也可以是一个两个元素的矢量，其中矢量的第一个元素表示最小步长的大小，第二个元素表示 SIMULINK 在给出错误提示之前最小步长误差提示出现的最大次数。初始步长通常由解法数据起始时间的导数值来确定，当仿真过程在起始阶段有较大变化时，则需要用户根据情况来确定合适的初始步长。

另外，SIMULINK 在 Solver 选项卡中还给出了误差限选项。它采用标准的当前误差技术来监控每一时间步的误差。误差限分为 Relative tolerance（相对误差限）和 Absolute tolerance（绝对误差限）。相对误差限是指误差相对于状态的值，它是一个比值，默认值是 1e-3。绝对误差限是表示误差值的阈值，即状态为零时，可以接受的误差。

◆ **Output options（输出选项）选项区域**

Solver 选项卡的输出选项允许用户设置 SIMULINK 产生的输出，它包括三个选项：Refine output（细化输出）、Produce additional output（产生额外输出）和 Produce specified

output only(只产生指定的输出),用户可以根据不同的需要进行设置。

2)Workspace I/O 选项卡

当单击 Workspace I/O 标签时,将打开 Workspace I/O 选项卡,如图 2-15 所示。该选项卡的作用是管理模型从 MATLAB 工作空间的输入和输出。



图 2-15 Workspace I/O 选项卡

从图 2-15 可以看到,该选项卡分为三个选项区域:Load from workspace(从工作空间加载)、Save to workspace(保存到工作空间)以及 Save options(保存选项)。下面对它们分别进行介绍:

◆ **Load from workspace 选项区域**

SIMULINK 的输入数据可以从 MATLAB 的工作空间获取。选中 Input 复选框,在文本框中输入指定的变量,然后进行参数修改,则 SIMULINK 可以在运行仿真时从仿真模型的 MATLAB 工作空间把输入提供到模型的顶层输入端口。外部输入采用的形式包括以下几种:

- 外部输入矩阵:默认外部的输入为[t u]。若从外部输入矩阵,此矩阵第一列必须是时间矢量且按升序排列,其他列则需指定输入值。输入矩阵的列数为n+1,其中 n 表示进入模型输入端口的信号总数,若[t u]在 MATLAB 工作空间已经定义,则指定此模型中的外部输入时只需选中 Input 复选框即可。
- 外部输入结构数组:SIMULINK 可以从 MATLAB 工作空间中以结构数组的形式读取数据,结构数组的名称在 Input 文本框中指定,并且要包括时间项和信号项。时间项包括仿真时间的列向量,信号项包括对应每个输入端口的子结构数组。

例 2-2 以图 2-16 所示的双输入模型为例,在 MATLAB 工作空间定义模型输入矢量 a。

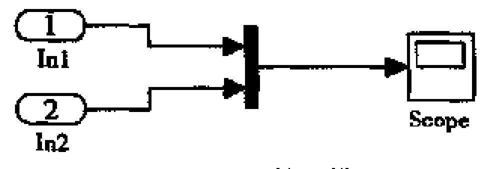


图 2-16 双输入模型

```
>> a.time=(0:0.1:10);
>> a.signals(1).values=sin(a.time);
>> a.signals(2).values=cos(a.time);
```

这时，指定外部输入只需要选中 Input 复选框并在后面的文本框中输入 a 即可。若 a 的时间为空，即 a.time=[ ]，此时 SIMULINK 会在每个时间步长内读取相应的输入端口的数据。

- 外部输入时间表达式：时间表达式可以是任何 MATLAB 表达式，SIMULINK 会在仿真进行时对表达式进行计算并将结果赋值给模型窗口。

◆ **Save to workspace 选项区域**

在 Save to workspace 选项区域中，可以设置保存到工作空间的变量，其中包括时间、状态量和输出。用户可以对保存到工作空间中的变量赋予不同的变量名称。模型的状态量和输出量可以是矩阵和结构数组，用法和前面介绍的外部输入一样。另外，在这个选项区域中还可以设置 Time、States、Output 和 Final state 等选项。其中，Time 和 Output 两个复选框默认是被选定的，因此，一般在模型运行之后，MATLAB 工作空间都会增加两个变量 tout 和 yout。对于 States 和 Final state 两个选项的含义读者在学习有关 S 函数的知识后就会十分清楚，这里不再赘述。

◆ **Save options 选项区域**

Save options 选项区域中包括的选项用来说明返回变量的格式和输出数据的个数。其中 Limit data points to last 复选框用于设置是否限制数据变量的点数，当选中该复选框后，其后的文本框将变为可用，在其中可以指定具体的点数值。Dicimation 文本框用于指定一个亚采样因子，它的默认值是 1，也就是说对每一个仿真时间点产生的值都保存。Format 文本框用来指定返回数据的格式，可以选择的格式有矩阵和结构数组。

3)Diagnostics 选项卡

单击 Diagnostics 标签，将打开 Diagnostics 选项卡，如图 2-17 所示。在此选项卡中允许用户设置 SIMULINK 在仿真过程中显示出错信息的等级。Diagnostics 选项卡可分为：

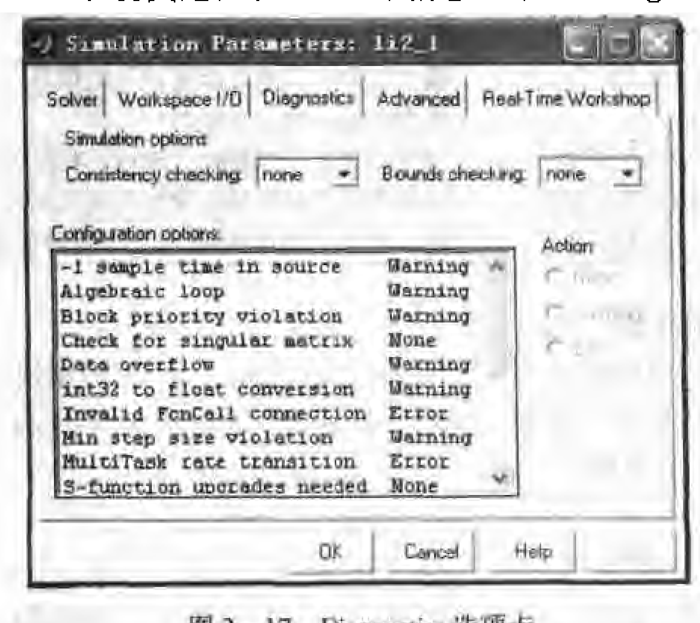


图 2-17　Diagnostics 选项卡

Simulation options(仿真选项)、Configuration options(配置选项)和 Action(处理方式)等三个选项区域。下面对各个选项区域进行介绍。

◆ **Simulation options 选项区域**

在 Simulation options 选项区域中包括 Consistency checking(一致性检验)和 Bounds checking(边界检验)两个选项,它们的默认值均为 auto。Consistency checking 是一种对 SIMULINK 的求解解法所作的某种假设检验的调试工具,它的主要作用是确认外部函数与 SIMULINK 内部模块是否遵循同样的规则和确认对于给定的时间 t 模块是否产生不变的输出。Bounds checking 主要用于检测仿真运行中是否对分配给它之外的空间进行操作。一般来说,应该尽量避免使用这种检验,它会降低仿真速度,但是对于检测模型中含有用户自己编写的 S 函数是否存在错误时,这种检验又是十分必要的。

◆ **Configuration options 选项区域**

Configuration options 选项区域用于处理在仿真过程中发生的一些非正常事件,常见的非正常事件类型以及意义如表 2-2 所示。对不同的非正常事件类型,用户可以在 Action 选项区域中选择不同的处理方式。

表 2-2　非正常事件说明

事件类型	说　明
-1 SAMPLE TIME IN SOURCE	设置源模块的采样时间为-1
ALGEBRAIC LOOP	在仿真运行过程中检测到代数环
CHECK FOR SINGULAR MATRIX	检测到奇异矩阵
DATA OVERFLOW	超出数据类型所能表示的范围以至于数据溢出
INT32 TO FLOAT CONVERSION	把 32 位整数转化为浮点数,这种转换会使精度损失
MIN STEP SIZE VIOLATION	下一个仿真时间步小于模型设定的最小步长,这可能发生在设定的误差限比最小步长小的仿真时间步中
MULTITASK RATE TRANSITION	运行在多速率模式下的两模块间进行无效的速率转换
S FUNCTION UPGRADES NEEDED	遇到具有使用当前版本的 S 函数特性的模块
SIGNAL LABEL MISMATCH	遇到具有相同的源信号,但具有不同的标签的虚拟信号
SIGNAL TASK RATE TRANSITION	运行在速率模式下的两模块间发生速率转换
UNCONNECTED BLOCK INPUT	模型中含有没有连接输入端的模块
UNCONNECTED BLOCK OUTPUT	模型中含有没有连接输出端的模块
UNCONNECTED LINE	模型中含有没有连接的信号线
UNNEEDED TYPE CONVERSION	含有不需要类型转化而进行了转换的模块
VECTOR/MATRIX CONVERSION	向量和矩阵间的转换发生在输入端口
BLOCK PRIORITY VIOLATION	运行仿真时,检测到模块优先级设置错误

◆ **Action 选项区域**

在 Action 选项区域中包括 None、Warning 和 Error 三个单选按钮,它们用于对不同的非正常事件类型进行不同的处理。其中:None 表示不进行任何提示;Warning 表示给出警告信息;Error 表示给出出错信息。

4)Advanced 选项卡

单击 Advanced 标签,将打开 Advanced 选项卡,如图 2-18 所示。在该选项卡中可以进行仿真参数的高级设置,这是 SIMULINK 4.0 版本新增加的功能。Advanced 选项卡

可分为Model parameter configuration(模型参数配置)、Optimizations(优化配置)和Action(处理方式)等三个选项区域。下面对各个选项进行介绍。

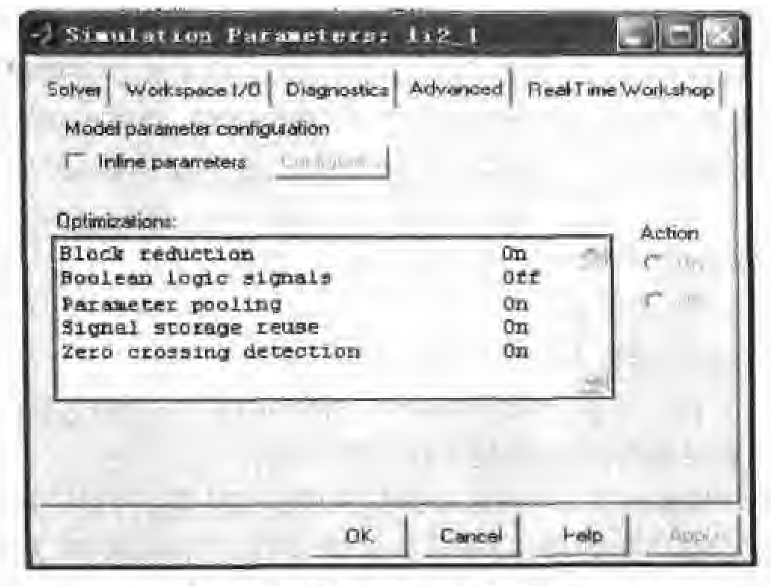


图2-18 Advanced选项卡

◆ **Model parameter configuration 选项区域**

在Model parameter configuration选项区域中,用户可以选定Inline parameters复选框,这样除了用户设置的特别参数之外,所有的参数都将被设置为不调谐,即SIMULINK在仿真过程中把它们当成常数处理,这样可以加快仿真速度。若用户想将某一变量保留成可调谐的,则可以单击Configure按钮,在弹出的仿真模型参数结构窗口中进行设置。

◆ **Optimization 选项区域**

Optimization选项区域中共包括5个选项。

- Block reduction:表示用一个合成模块来代替一组模块,以提高仿真速度。
- Boolean logic signals:表示允许输入双精度型数据的模块,同时也能输入布尔数据类型。
- Parameter pooling:用于代码生成,在不进行代码生成时这个选项保持on状态即可。
- Signal storage reuse:表示允许存储空间再利用,这样可以提高仿真速度。一般情况下,在模型调试过程中不允许存储空间再利用。
- Zero crossing detection:用于设置是否进行过零点检测。

◆ **Action 选项区域**

在Action选项区域中包括On和Off两个单选按钮,它们用于是否进行优化处理。其中,On表示进行优化处理,Off表示不进行优化处理。

2. 运行仿真

在完成仿真参数的设置后,就可以开始运行仿真了,除了使用命令和窗口两种方法仿真外,还可以按下Ctrl+T组合键运行仿真。

在仿真运行过程中,如果打算停止仿真运行过程,在“仿真模型”窗口的菜单栏中执行

Simulation/Stop 命令，或按下 Ctrl + T 组合键即可。

3. 诊断对话框

如果在仿真运行过程中有错误发生，SIMULINK 会中断仿真并弹出仿真诊断对话框来显示错误信息，例如，运行如图 2-19 所示的仿真模型时，会弹出如图 2-20 所示的诊断对话框。诊断对话框分为上下两部分，上部显示所有的错误信息，主要包括以下几类信息：

- Message：错误信息类型（例如模块错误、警告等）。
- Source：导致错误的模型元素（模块或是信号线）的名称。
- Fullpath：导致错误的模型元素的路径。
- Reported By：报告错误来源组件（例如 SIMULINK、STATEFLOW 等）。
- Summary：错误消息的概括，便于在列表中显示。

诊断对话框的下部显示当前所选定错误信息的详细内容。另外，通过双击对话框上部的错误信息，或双击错误源的名称，或单击墩话框中的 Open 按钮，都可以显示所选定错误信息的具体位置，同时打开模型并用黄色来表示出现错误的模块或信号线。

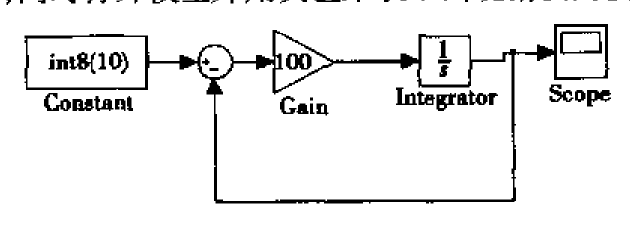


图 2-19　仿真模型

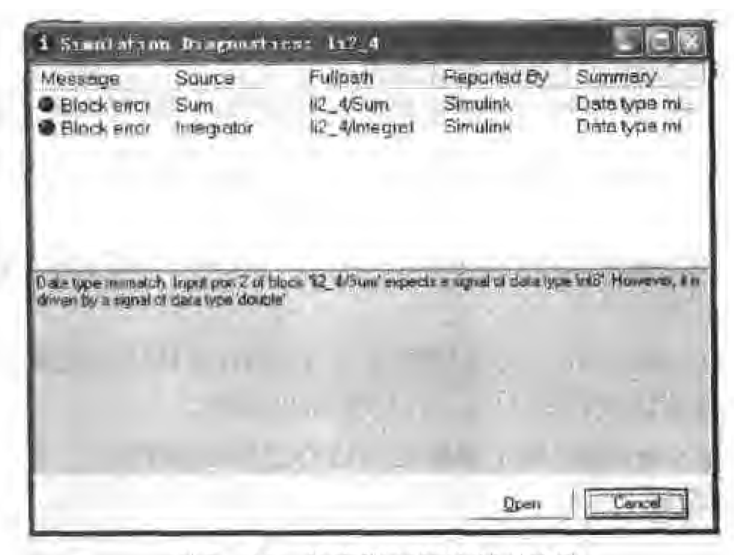


图 2-20　仿真模型的诊断对话框

2.3　使用命令进行仿真

与使用菜单命令运行仿真相比较，使用命令行运行仿真时用户可以运行 M 文件，这

样允许仿真和模块的参数不断地被改变。另外，在进行 Monte Carlo 分析时，用户可以随机地修改参数和在每一个循环中运行仿真，以得到不同的结果。本节将介绍常用的运行系统仿真的命令，随后介绍如何提高系统仿真的速度和精度。

2.3.1　仿真命令

用户可以在命令行输入 sim 和 set _ param 命令来运行系统仿真。

1. 使用 sim 命令运行仿真

MATLAB 提供了 sim 命令，使用户可以在 MATLAB 环境下以命令行或是 M 文件的形式运行 SIMULINK 仿真模型。sim 命令使在不同的参数值、不同的输入值和不同的初始条件下反复运行仿真变得十分简单方便，而且可以很容易地获得分析时所需要的状态量。sim 命令通常与 simset、simget 命令一起配合使用。

1)sim 命令

调用 sim 命令的格式是：

```
[T,X,Y] = SIM('model',TIMESPAN,OPTIONS,UT)
```

或

```
[T,X,Y1,...,Yn] = SIM('model',TIMESPAN,OPTIONS,UT)
```

其中，等式右侧的参数中只有 model 是必须的，而其他参数都可以设置为空(即[ ])。另外，模型中所设置的参数将会代替在仿真参数对话框中设置的参数值。接下来具体说明命令中各个变量和参数的意义。

- T：返回仿真时间矢量。
- X：返回仿真状态矩阵或结构形式，排列的顺序是先连续状态，后离散状态。
- Y：返回仿真输出矩阵或结构形式，其中的每一列对应一个根层次的输出端口。
- Y1，…，Yn：返回模型中的根层次输出端口的输出，输出端口共有 n 个。
- Model：要运行的 SIMULINK 模型的名称。
- TIMESPAN：用于指定仿真的起始时间和结束时间。它的说明方式有以下四种：
  设置为空(即[ ])表示使用仿真参数设置对话框中的参数值；设置为 Tfinal 表示仅仅指定仿真结束时间，而仿真起始时间由仿真参数设置对话框中的参数值确定；设置为[TStart TFinal]表示指定了仿真参数的起始和结束时间；设置为[TStart OutputTimes TFinal]表示除了指定了仿真起始和结束时间外，还会产生附加的时间点。
- OPTIONS：它的结构由 simset 命令创建、更新及显示，用于指定可选的仿真参数，这些可选的仿真参数将在以后章节中作具体介绍。
- UT：可选的外部输入。它可以是一个输入变量表或一个 MATLAB 函数的函数名。若它是一个输入变量表，其形式必须是[t ,ut1 ,ut2 ,…]。

当由 sim 命令输入的变量代替了仿真参数时，模型本身并不发生变化。因此 sim 命令只有在运行仿真时才有效，当仿真结束后，模型参数又恢复到 sim 命令执行前的状态。

例如:

命令:>> [t,x,y]=sim('li2_1')

这个命令是运行"li2_1"模型的仿真,并且仿真参数值均采用仿真参数设置对话框中的默认值。

命令:>> [t,x,y]=sim('li2_1',[0,5],simset('debug','on'))

这个命令的功能是对"li2_1"模型进行仿真调试,并且仿真时间范围是从 0~5。

2)simset 命令

simset 命令是用来创建和编辑 options 结构的,此命令的调用方式共有三种。

调用方式一:

>> options=simset('name1',value1,'name2',value···)

其中 name1、value1 等表示属性名和属性值,具体的属性名和属性值如表 2-3 所示。

表 2-3 属性名和属性值说明表

属性名	属性值及其意义说明
SOLVER	微分方程的求解解法选择
RELTOL	相对误差限,默认值为 1E-3
ABSTOL	绝对误差限,默认值为 1E-6
REFINE	指定细化输出点,其值必须是正整数,默认值是 1,只适于变步长解法
MAXSPEP	最大积分步长,默认值为 AUTO
INITIALSTEP	初始积分步长,默认值为 AUTO
MAXORDER	最大积分阶次,这个属性仅在使用 ODE15S 解法时需要设置
FIXEDSTEP	固定步长大小,当使用固定步长解法时需要设置
OUTPUTPOINTS	设置输出点类型,其值为'SPECIFIED'或'ALL','SPECIFIED'表示输出所指定的时间点上的值,'ALL'表示输出所有积分点上的值
OUTPUTLARIABLES	指定输出变量的值,其值为'T'、'Y'、'X'的组合,默认值为 TXY
MAXROWS	指定输出矩阵的最大行数,默认值为 0
DECIMATION	设置输出点间隔,其值必须是正整数。例如为 1 则表示输出所有点
INITIALSTATE	设置模型状态变量的初始状态
FINALSTATENAME	设置需要保存模型状态终值的变量名
TRACE	设置跟踪信息,其值为'MINSTEP'、'SIMINFO'或'COMPILE',默认值是
SRCWORKSPACE	选择对 MATLAB 表达式求值的工作空间,默认值为'BASE'、'CURRENT'或'PARENT'
DSTWORKSPACE	选择对变量赋值的工作空间,默认值为'BASE'、'CURRENT'或'PARENT'
ZEROCROSS	过零点检测开关,其值为'ON'或'OFF',默认值为'ON'

调用方式二:

>> options=simset(oldopt_struct,'name1',value1,...)

其中 oldopt_struct 是已经存在的 options 结构,此命令的功能是用来修改 oldopt_struct 结构中的属性值。

调用方式三:

>> options=simset(oldopt_struct,newopt_struct)

其中 oldopt_struct 和 newopt_struct 是已经存在的 options 结构,newopt_struct 享有优先权,此命令的功能是 newopt_struct 中定义的属性将代替 oldopt_struct 中定义的

相同属性。

另外,还可以使用不带任何参数的单独的 simset 命令,如:

```
>> simset
```

这个命令将列举出所有属性的名称及其它们的值。

3)simget 命令

simget 命令用来获得整个 options 结构或其中指定的某个属性值。

```
>> opt_struct=simget(model)
```

这个命令用于获得当前整个 options 结构,options 结构由 sim 或 simset 命令定义。

```
>> value=simget(model,name)
```

simget 命令的作用是获得指定的仿真参数或是求解器的属性值。

2. 使用 set_param 命令运行仿真

set_param 命令可以用来启动、暂停、停止或是继续运行一个仿真,也可以用来更新模型模块图;还可以用 get_param 命令来检查一个仿真状态。set_param 命令运行仿真的调用格式如下:

```
>> set_param('object','simulationcommand','cmd')
```

其中,object 是系统或是模型的名称 simulationcommand 是仿真命令;cmd 是具体的控制仿真命令的取值,它可以是'start'、'pause'、'stop'、'continue'或'update'。选择不同的控制命令值可以实现不同的功能。

用 get_param 命令检查仿真状态时,调用的格式为:

```
>> get_param('object','simulationstatus')
```

这时,命令的返回值是:'stopped'、'running'、'initializing'、'terminating'或'external'。

2.3.2 仿真性能与精度提高

仿真的性能与精度受到很多因素的影响,其中包括模型的设计和仿真参数的选择。对于大多数问题,使用默认的仿真参数值求解解法可以精确而有效地解决,但是对于某些模型,如果能够适当地调整求解解法和仿真参数就可以得到更好的仿真结果。而且,如果用户熟悉仿真模型的性能,并将这些信息提供给求解器,得到的仿真结果的性能和精度将会更好。

1. 加快仿真速度

影响仿真模型仿真速度的因素有很多,常见的因素包括:

- 模型中包含 MATLAB Fun 模块。当模型中包含 MATLAB Fun 模块时,SIMULINK 会在每一个仿真时间步调用 MATLAB 解释器,这样就会使仿真速度大大降低。所以应该尽可能地使用 SIMULINK 内置的 Fun 模块或最基本的数学(math)函数。
- 模型中包含 M 文件形式的 S 函数。当模型中包含 M 文件形式的 S 函数时,SIMULINK 会在每一个仿真时间步调用 MATLAB 解释器,这将会大大降低仿

真速度,为了提高仿真速度,可以考虑将M文件形式的S函数转换为等价的子系统或是C-MEX形式的S函数。

- 模型中包含Memory模块。使用Memory模块时会使变阶求解器(如ode15s解法)在每个仿真时间步将阶数重设为1阶。
- 最大的仿真时间步长过小。如果曾经更改过最大仿真时间步长,解决的办法是把最大仿真时间步长参数设置回SIMULINK的默认值auto。
- 对仿真精度要求过高。一般来说相对误差限设置为0.1%就足够了。对于含有取值会趋向于零状态的模型,进行仿真时如果绝对误差限太小,仿真时状态值在零点附近会花费较多的时间步,这样就会使仿真速度降低。
- 仿真时间过长。这时为了提高仿真速度可以适当地减小时间间隔。
- 如果要解决的是一个刚性问题,而采用的是非刚性的求解解法,这时可以试一试ode15s解法是否有效。
- 模型使用的采样时间相互之间没有倍数关系,这样混合的采样时间会导致求解器采用太小的时间步长来保证采样时间符合所有的采样时间要求。
- 仿真模型中包含代数环。代数环的求解解法是在每一个时间步反复地进行计算,这样会大大降低仿真性能。
- 模型把Random Number模块的输出作为Integrator模块的输入。对于连续系统解决办法是使用Source模块库中的Band-Limited WhiteNoise(白噪声)模块。

2. 提高仿真精度

检验仿真精度的方法是修改仿真的相对误差限和绝对误差限。仿真运行一段时间之后,减小相对误差限或是绝对误差限,并重新运行仿真。同时比较两次仿真的结果是否有较大的差别,如果它们之间的差别不大,则表示所求得的解是收敛的。

如果仿真运行了一段时间之后,结果变得不稳定,原因可能是:

- 系统本身可能是不稳定的。
- 如果求解方法使用的是ode15s解法,可以把最大阶数设定为2阶(稳定的最大阶数)或者采用ode23s解法。

如果仿真结果不很精确,原因可能是:

- 对于一个含有会趋向于零状态的模型,如果绝对误差限设置得太大,则会使仿真在接近零状态附近运行的仿真时间步太少,这样会导致仿真精度不高。解决的办法是减小绝对误差限或是在积分(Integrator)模块的对话框中调整每个状态的绝对误差限。
- 如果减小绝对误差限不能有效地提高仿真精度,可以试着减小相对误差限并减小仿真步长,以增加仿真步数。

2.4 仿真结果分析

仿真的主要目的是通过创建系统模型以得到某种计算结果。因此,仿真结果分析是

进行系统建模与仿真的一个重要环节。仿真结果分析有助于系统模型的改进以及性能的完善,同时仿真结果分析也是运行仿真的一个主要目的。不仅可以通过SIMULINK提供的输出模块进行仿真结果的分析,而且在MATLAB中也提供了一些用于进行仿真结果的分析的函数和命令。读者可以根据系统模型的要求和实际需要选择进行仿真结果分析的方法。

2.4.1 利用输出模块分析

输出模块可以用于显示系统的输出,即显示系统的仿真结果。利用SIMULINK输出模块进行仿真结果的输出有以下几种方法:

- 观察输出轨迹。
- 以文件形式输出。
- 直接显示数据输出。
- 用表盘和量计显示输出。
- 进行数字信号处理、分析输出。

1. 观察输出轨迹

利用输出模块进行输出结果分析最直观、最便捷的方法是观察输出轨迹。SIMULINK提供的观察输出轨迹的输出方法有3种:

1)信号输入到Scope(示波器)模块或XY Graph(相轨迹示波器)模块观察输出轨迹

使用Scope模块观察输出轨迹在前面已经使用过多次,这种方法较简单与常用,用这种方法绘制的输出轨迹是信号的时间曲线。利用XY Graph模块绘制的输出轨迹是两路输入信号的对比图。例如两路输入信号分别为sin(t)和sin($2t$),建立如图2-21所示的仿真模型,使用XY Graph模块观察输出轨迹,将得到如图2-22所示的轨迹。使用Scope模块或XY Graph模块观察输出轨迹是十分简单的,无需在仿真结束后附加任何仿真命令就能得到仿真结果。

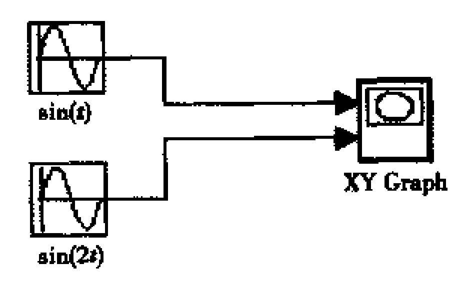


图2-21 使用XY Graph模块观察输出轨迹模型

2)将输出信号写入返回变量,并利用MATLAB命令绘图

这种方法可以在仿真结束后,在MATLAB工作空间自动生成两个变量tout和yout,分别返回时间矢量和各个输出端子的仿真结果。然后利用返回变量tout和yout,并利用MATLAB的绘图命令显示并标注仿真输出轨迹。当然,使用这种方法之前首先需要在仿

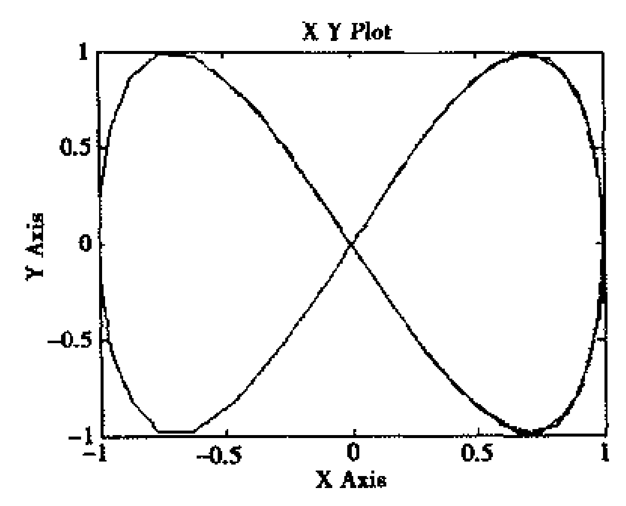


图2-22　使用XY Graph模块观察得到的输出轨迹

真参数设置对话框的Workspace I/O选项卡中选中变量时间(Time)和输出(Output)复选框，再对相应的时间变量(tout)和保存输出变量(yout)进行设置。

下面以如图2-23所示的'li2_1'模型中使用返回变量观察输出轨迹为例进行介绍。首先，利用以前介绍的方法进行仿真参数的设置，特别是要在Workspace I/O选项卡中选中变量时间(Time)和输出(Output)复选框；然后，设置相应的时间变量(tout)和保存输出变量(yout)。运行仿真后，利用MATLAB的plot绘图命令就可以得到输出轨迹。调用MATLAB绘图命令的格式为：

```
>> plot(tout,yout)
```

运行该命令后将得到如图2-24所示的输出轨迹，它与使用Scope得到的输出轨迹是一样的。

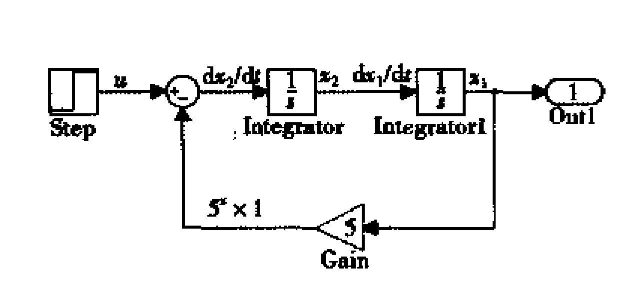


图2-23　使用返回变量的仿真模型

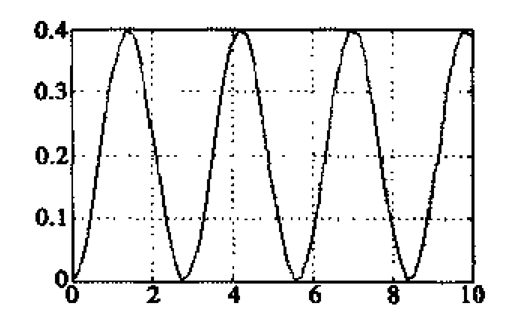


图2-24　使用返回变量观察输出轨迹

3)使用To Workspace模块把输出写入到MATLAB工作空间，然后通过MATLAB的绘图命令绘制输出轨迹

To Workspace模块能够接受矢量输入，在返回工作空间的变量中，每一个输入元素的轨迹都保存在一个列向量中。

例如，在图2-23所示的仿真模型中，把返回变量模块(Out1)用To Workspace模块代替，如图2-25所示。然后，在To Workspace模块的参数设置对话框中设置返回的Variable name(变量名称)和Save format(变量的存储格式)，To Workspace模块参数设置对话框如图2-26所示。

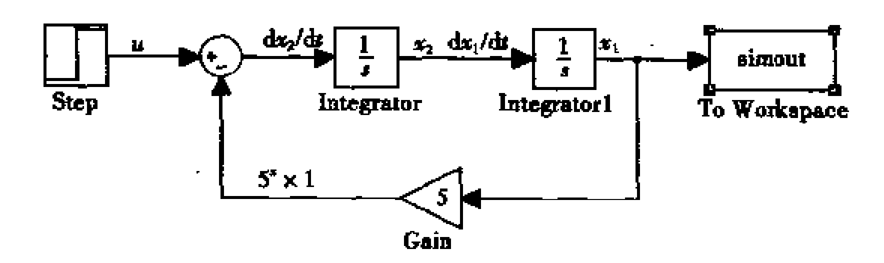


图 2－25　使用 To Workspace 模块的仿真模型



图 2－26　To Workspace 模块参数设置对话框

在完成参数设置并运行仿真之后，再执行下面的 MATLAB 仿真命令：

```
>> simout
simout =
          time: []
       signals: [1x1 struct]
     blockName: 'li2_7/To Workspace'
```

然后，执行下面的绘图命令：

```
>> plot(tout,simout.signals.values)
```

这时就会得到如图 2－24 所示的仿真输出轨迹。

2. 以文件形式输出仿真结果

利用输出模块库中的 To file 模块可以将仿真结果以 Mat 文件的格式直接保存到数据文件中。另外，在 To file 模块的参数设置对话框中可以设置需要保存的文件名。

3. 直接显示数据输出

SIMULINK 在输出模块库中提供了一个 Display 模块，它用于直接显示数据。它以数值格式显示的形式连接到信号上，可以同时显示多路信号。

注意：由于仿真是非实时的，对一般问题来说，仿真过程是非常快的，所以不适用于连接数值显示模块来观察输出结果。

4．表盘和量计显示输出

利用表盘和量计显示输出结果时，需要安装 Dials & Gauges 模块库。仿真结果用表盘和量计显示输出方式类似于过程控制现场用于显示信号的各种各样的仪表，它是 SIMULINK 提供的一组基于 ActiveX 技术的显示部件。用户可以通过 SIMULINK 直接进入 Dials & Gauges(表盘和量计)模块库，或在 MATLAB 命令窗口输入 dnglib 命令进入 Dials & Gauges(表盘和量计)模块库。

5．数字信号处理、分析输出

SIMULINK 允许在一些信号的后面直接连接数字信号处理模块，以便获得信号的相关航速、功率谱分析或快速 Fourier 变换等数字信号处理结果。在 DSP Blockset(数字信号处理模块库)中提供了丰富的数字信号处理模块，如图 2－27 所示。另外，SIMULINK Extra(其他模块库)中的 Additional Sinks(附加输出模块库)也提供了一些输出模块，如图 2－28 所示。

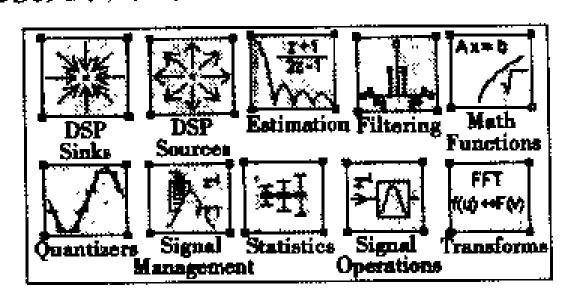


图 2－27　DSP Blockset

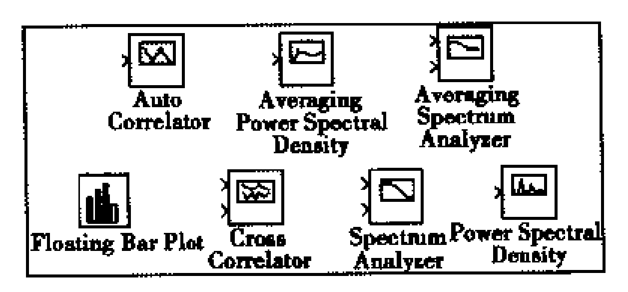


图 2－28　Additional Sinks

例 2－3　对输入信号 $\sin(t)+\sin(2t)$ 进行信号的功率谱分析。

此例建立的仿真模型如图 2－29 所示。仿真后得到的信号的功率谱分析图形如图 2－30所示。

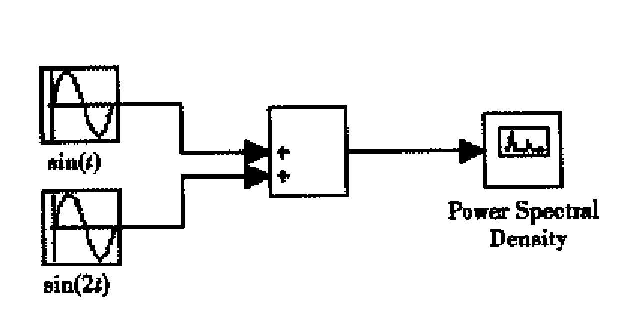


图 2－29　进行功率谱分析的仿真模型

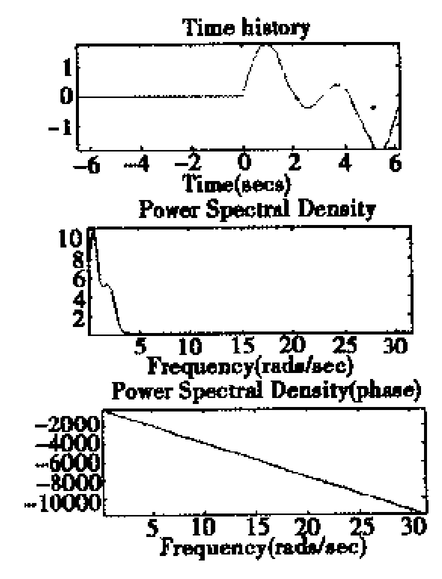


图 2－30　功率谱分析的仿真结果

2.4.2　使用函数分析

前面介绍了利用 SIMULINK 提供的输出模块进行系统仿真结果的分析，可以看出，使用这种方法进行仿真结果分析操作十分简便，显示十分直观。同时，MATLAB/SIMULINK 也提供了用于仿真结果分析的函数，使用函数进行仿真结果分析的优点是灵活方便。可用于仿真结果分析的函数很多，这里只介绍部分常用的仿真结果分析函数以供读者参考。

1. 利用 Max/Min 函数分析仿真结果

Max/Min 函数是用于求最大/最小值的函数，它在进行系统优化或控制分析时非常有用。由于 Max 函数和 Min 函数的调用格式是一样的，所以这里只对 Max 函数进行详细介绍。Max 函数的调用格式为：

```
Max(X)
Max(X,Y)
[Y I]=Max(X)
[Y I]=Max(X,[],dim)
```

Max(X)返回的是 X 之中的最大值，当 X 是一个矢量时，它将返回最大元素的值；当 X 是一个矩阵时，它的返回值是一个行向量，行向量中每一个元素都是 X 每一列中的最大元素值。Max(X,Y)则返回一个与 X 和 Y 具有相同维数的向量或矩阵(这里要求 X 和 Y 也具有相同维数)，返回值的每一个元素都是对应的 X 和 Y 中的最大元素值。

例 2-4　当 x=[1 4;2 3;5 6],y=[2 3;7 8;9 0]时，则：

```
>> max(x)
ans =
    5    6
>> max(x,y)
ans =
    2    4
    7    8
    9    6
```

[Y I]=Max(X)命令不仅返回 X 中的最大值，而且返回最大值所在的位置。[Y I]=Max(X,[],dim)命令中指定了求最大值时开始的维数 dim，这个命令将从指定的维数开始求 X 中的最大值。

例 2-5 当 x=[1 4;2 3;5 6],y=[2 3;7 8;9 0]时，则：

```
>> [y i]=max(x)
y =
    5    6
i =
    3    3
>> [y i]=max(x,[],2)
y =
    4
```

```
        3
        6
i =
        2
        2
        2
```

下面以一个简单的例子说明 Max 函数在控制系统分析仿真结果中的应用。一个二阶控制系统的仿真模型如图 2－31 所示。

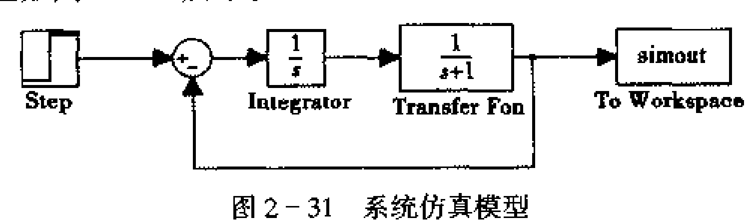


图 2－31　系统仿真模型

在自动控制系统时域仿真结果分析中，一个比较重要的指标就是超调量，它是系统响应超出终值的量，通常用百分量来表示。对于图 2－31 所示的系统，它的超调量只需以下命令即可求出：

```
>> yout = simout.signals.values;
>> sigma = (max(yout) - yout(length(tout)))/yout(length(tout)) * 100
sigma =
  16.0099
```

像 Max 函数这样的可以用于仿真结果分析的函数还有很多，例如 Min、Median、Sort、Mean 等等，读者可以参阅本书附录，其具体用法可以从帮助文件中得到，这里不再赘述。

2. 平衡点的确定

在进行非线性系统分析过程中，利用仿真结果分析、评估系统的稳定性或稳定状态时往往需要用到系统的平衡点（Equilibrium points）。系统的平衡点是指系统所有状态等于零的点。使用 SIMULINK 提供的 trim 函数进行平衡点的求取十分方便。通过调用 trim 函数能够找到满足特定输入、输出和特定条件的平衡点，以及系统状态导数指定为非零值的点。trim 函数的具体调用格式如下：

```
>> [X,U,Y,DX,OPTIONS] = TRIM('SYS',X0,U0,Y0,IX,IU,IY,DX0,IDX,OPTIONS,T)
```

其中，在所有的输入输出变量中只有‘SYS’是必须的，它表示需要求取平衡点的系统的名称，而其他的各个变量可以根据具体需要进行设置。其他变量的含义如下：

- X0，U0，Y0：开始进行搜索的（X，U，Y）点的状态、输入和输出的推测值。
- IX，IU，IY：在进行约束搜索时用于指定 X0，U0，Y0 中保持不变的分量的下标。
- DX0，IDX：这两个参数是在指定系统状态导数为非零值时用到的，它们需要配合使用。DX0 用于指定哪些状态的导数非零，IDX 用于指定状态非零导数的具体值。
- OPTIONS：用于进行优化算法的参数选项设置。该参数使用 trim 函数把优化算法的参数选项设置传递到优化参数中来查找平衡点。
- T：用于指定时变状态导数的具体计算时刻。

下面以一个简单的例子说明利用 trim 函数进行平衡点求取的方法。

例 2-6　系统的仿真模型如图 2-32 所示，使用 trim 函数求使两个输出分别为 1 和 2 的状态值和输入值。

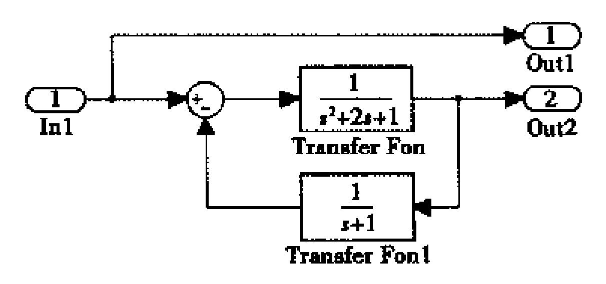


图 2-32　系统仿真模型

首先，设置状态 x 和输入 u 的初始估计值以及输出的期望值：

```
>> x0=[0 0 0]';
>> u0=0;
>> y0=[1 2]';
```

然后，指定哪一个变量保持不变，哪一个变量是变化的。

```
>> ix=[];
>> iu=[];
>> iy=[1,2];
```

最后，通过调用 trim 函数即可求得所需的平衡点。

```
>> [x,u,y,dx]=trim('li2_15',x0,u0,y0,ix,iu,iy)
x =
     0.0000
     1.0000
     1.0000
u =
     2.0000
y =
     2.0000
     1.0000
dx =
   1.0e-015 *
     -0.3331
     0.0000
     0.3331
```

2.5　子系统

随着研究的系统越来越大，越来越复杂，直接使用基本的 SIMULINK 模块创建的系统模型会十分庞大，而且信号的传输方向也会变得十分不明显。为了简化模型以及增加

它的可读性，可以将一些具有独立功能的模块划分成一组，构成子系统。SIMULINK 环境下的子系统就像 MATLAB 的 M 文件或 C 语言的子程序一样，创建十分方便。本节将首先系统地介绍子系统的创建和封装方法与技巧，然后通过应用实例来说明子系统模块的创建过程。

2.5.1 子系统的创建

使用子系统不仅可以使仿真模型中显示的模块数量减少，便于读图，而且还建立了层次化的仿真模型框图，子系统模块在一个层次，组成各个子系统的模块在另外一个层次。创建子系统的方法有两种，一种是利用模型中已经存在的模块创建子系统；另一种是通过向子系统模块加入新的模块创建子系统。下面以创建 PID 控制器为例分别介绍这两种创建方法。

1. 通过已有模块创建子系统

通过已有模块创建子系统比较简单，只需先选定组成子系统的所有模块，然后在仿真模型窗口的菜单栏中执行 Edit/Create subsystem 命令即可。下面以通过已有模块创建一个 PID 控制器子系统为例，介绍通过已有模块创建子系统的过程和方法。

PID 控制器在自动控制系统中应用比较广泛，它的仿真模型如图 2－33 所示。在工程中，它的数学模型为：

$$K_p * \left(1 + \frac{1}{T_i * s} + \frac{T_d * s}{\frac{T_d}{N} * s + 1}\right)$$ 其中，一般要取 $N \geq 10$。

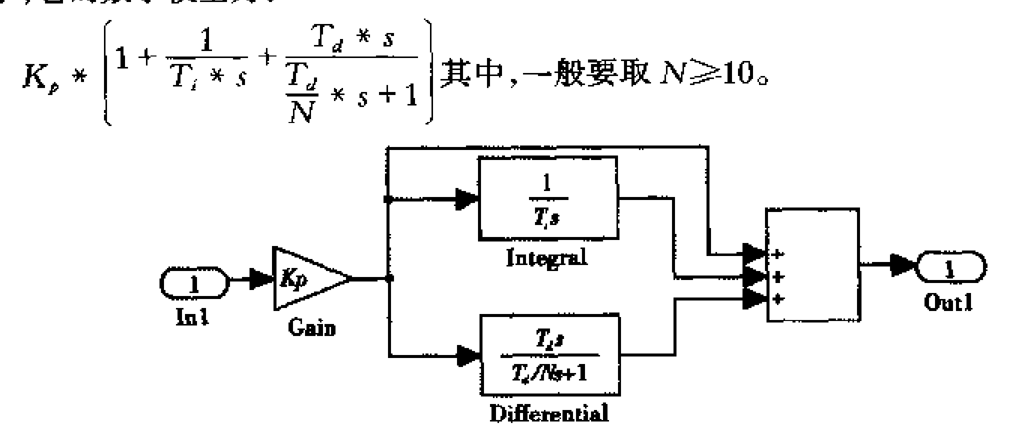


图 2－33 PID 控制器仿真模型

通过已有模块创建 PID 控制器子系统的具体步骤如下：

(1)选中组成 PID 控制器的所有模块。

(2)在仿真模型窗口的菜单栏中执行 Edit/create subsystem 命令，则会把已经选中的模块装入一个名为 subsystem 的模块中，如图 2－34 所示。

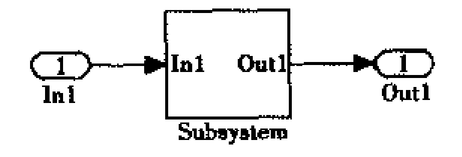


图 2－34 创建 PID 子系统之后的仿真模型

这样即完成了 PID 控制器子系统的创建。接下来可以更改子系统模块的名称。具体的操作方法是：双击已经创建好的子系统模块打开子系统，在打开的子系统模块窗口重新命名输入输出端口的名称。例如，把 PID 控制器子系统模块名称改为 PID Controller，而其输入输出端口名称改为 Input 和 Output，此时的模型如图 2－35 所示。

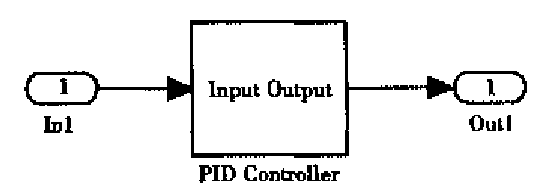


图 2－35　修改名称之后的 PID 子系统仿真模型

2. 通过子系统模块创建子系统

通过子系统模块创建子系统也十分简单，首先新建一个仿真模型窗口，在 SIMULINK 的 Subsystems(子系统)模块库中(4.0 以前版本在 Signals & System 模块库中)复制一个 Subsystem 模块到仿真模型窗口，然后双击该模块，将产生一个模型窗口，最后利用以前介绍的创建 SIMULINK 仿真模型的方法把子系统包含的所有模块复制到其中，并对其进行信号连接。

利用这种方法创建 PID 控制器子系统的具体过程如下：首先从模块库复制一个 Subsystem 模块，然后打开子系统模块，把 PID 控制器的所有模块复制到其中，最后进行信号线的连接。利用这种方法创建的子系统与通过已有模块创建的子系统的结果是一样的。

2.5.2　子系统的封装

利用前面介绍的方法创建子系统，达到了简化模型、提高模型可读性的目的。但是这样建立的子系统要从 MATLAB 工作空间直接获取变量，与 SIMULINK 模块库中的基本模块还有很大的差别，存在着不少缺陷。因此，需要对所创建的子系统进行封装，进一步完善子系统。所谓封装技术就是将对应的子系统的内部结构隐藏起来。这样在访问此子系统模块时就只出现一个参数设置对话框，只要将所需要的变量参数输入到对话框中即可。实际上，SIMULINK 的基本模块也是子系统封装后得到的，所以子系统封装之后与基本模块是一致的。例如，前面用到的增益模块或是传递函数模块等，它们的内部结构是不可见的，使用时只需要在参数设置对话框中输入相应的参数即可。

子系统的封装过程如下：

(1)选中已创建的子系统，在仿真模型窗口的菜单栏中执行 Edit/Mask subsystem 命令，将会打开 Mask Editor(封装编辑器)对话框，如图 2－36 所示。

(2)利用封装编辑器可以实现子系统对话框的设计。需要设计的内容主要包括子系统的图形标识、变量参数以及模块描述和帮助信息。

(3)关闭封装编辑器，则得到了新建的封装之后的子系统模块。



图 2－36　子系统封装编辑器对话框

1. 封装编辑器

在介绍子系统的详细封装过程之前，首先需要介绍封装编辑器。封装编辑器窗口包括 Icon(图形标识)、Initialization(初始化)和 Documentation(文档)等三个选项卡。单击相应的标签将打开与之对应的选项卡。

1) Icon 选项卡

打开的 Icon 选项卡如图 2－36 所示。Icon 选项卡包括以下选项：

- Mask type(封装类型)文本框：在该文本框中用户可以任意填写，它接受中、英文输入。
- Drawing commands(图形标识命令)文本框：这个文本框的作用是对封装后的子系统进行图形标识。用户可以使用 MATLAB 的 plot 绘图命令绘制图形，可以使用 disp 命令在子系统的图标上写入字符串名称，可以使用 image( )函数进行图片显示。
- Icon frame(标识边框)下拉列表框：用于设置模块图标是否有边框，它的值可以设置为 Visible(可见的)和 Invisible(不可见的)。默认值是 Visible，即显示图标边框。
- Icon transparency(标识透明)下拉列表框：通过它可以将图标设置为 Opaque(不透明的)或 Transparent(透明的)，默认值是 Opaque，这时图标上的图形会将子系统模块的端口信息覆盖。如果要显示端口信息，将此属性值设置为 Transparent 即可。
- Icon rotation(标识旋转)下拉列表框：通过它可以将图标设置为 Fixed(固定的)或 Rotates(旋转)，默认值为 Fixed，表示在旋转或翻转子系统模块时，图标不会随之旋转或是翻转。如果选择 Rotates，则表示在旋转或翻转子系统模块时，图标也会随之旋转或是翻转。
- Drawing coordinates(图形尺寸调整)下拉列表框：该下拉列表框用于设置绘图的尺度单位以及图标是否会随着模块大小的变化而变化。可以将其设置为 Pixels(像素点)、Autoscale(自动定标)和 Normalized(标准化)，默认的格式是Autoscale。

2)Initialization 选项卡

在进行子系统模块封装时,最重要的一步就是设计子系统模块变量参数设置对话框,这需要在封装编辑器的 Initialization 选项卡中完成。在这个选项卡中可以对封装子系统模块的参数设置对话框进行变量参数的提示与设置的设计。

Initialization 选项卡如图 2-37 所示。在该选项卡中也需要设置封装类型(Mask type),在 Documentation 选项卡中同样需要设置封装类型。值得注意的是封装类型只要在编辑器任何一个选项卡中设置后,SIMULINK 会自动将它的值显示于其他选项卡中。所以,只要在其中一个选项卡中设置了封装类型之后,在其他选项卡中就不需要再设置了。



图 2-37　Initialization(初始化)选项卡

在 Initialization 选项卡中,需要设置的最重要内容在模块参数设置对话框设计区。在这个选项卡中,单击 Add 或 Delete 按钮,就可以添加或删除变量。当单击 Add 按钮时,其右侧列表框中的"≪end of parameter list≫"下移一个位置,并且在其上方会出现一个空行。这时,就可以在 Prompt(提示)文本框中填写该变量的提示信息,所填写的提示信息将自动显示在空行位置。在 Variable(变量)文本框中填写变量名,在 Control type(控件类型)下拉列表框中选择子系统模块封装后参数设置对话框中变量值的输入方式,若选择默认值"Edit",则变量值会通过文本框输入。在 Assignment(标识)下拉列表框中选择输入变量的数据类型,其默认值"Evaluate"表示"数值类"的数值或计算结果是数值的表达式。最后对 Popup strings(列表字符串)文本框进行设置,这个文本框只有在 Control type 设置为 Popup 时才变为可用,否则将不可用。

在 Initialization 选项卡的最下端还有一个 Initialization commands(初始化命令)文本框,它用于定义生成封装子系统模块过程中所需要的变量。例如,封装子系统模块过程中如果使用 MATLAB 的 plot 绘图命令绘制模块图标,在此可以定义所需要的变量。

3)Documentation 选项卡

Documentation 选项卡是提供给用户编写封装子系统模块的性质说明和帮助信息的。Documentation 选项卡如图 2-38 所示。它包括 Mask type(封装类型)、Block description(模块说明)和 Block help(模块帮助)三个文本框。封装类型的设置在这里不再赘述。

Block description 文本框用于编写封装子系统模块的简要性质说明，它将显示在封装后子系统模块的参数设置对话框中。模块性质说明的编写允许使用中、英文。Block help 文本框用于编写封装子系统模块的帮助信息，当单击封装子系统模块参数设置对话框中的 Help 按钮时，该文本框中的内容就会显示出来。帮助信息的编写只允许使用英文。



图 2－38　Documentation（文档）选项卡

2. 封装实例

接下来，以对"PID 控制器子系统"模块进行封装来说明封装子系统的具体操作过程。

首先打开封装编辑器，在 Icon 选项卡中的 Mask type 文本框中填入 PID controller，则在封装编辑器的另外两个选项卡（Initialization 和 Documentation）的 Mask type 文本框中将会自动显示这个信息。在 Drawing commands 文本框中输入"disp（'PID \ncontroller'）"，对封装的子系统模块进行标识，其中\n 表示回车换行。再在 Icon frame 下拉列表框中选择"Visible"，即图标边框可见。在 Icon transparency 下拉列表框中选择"transparent"，在 Icon rotation 下拉列表框中选择"Fixed"，在 Drawing coordinates 下拉列表框中选择"Normalized"。完成设置之后，封装子系统模块如图 2－39 所示。

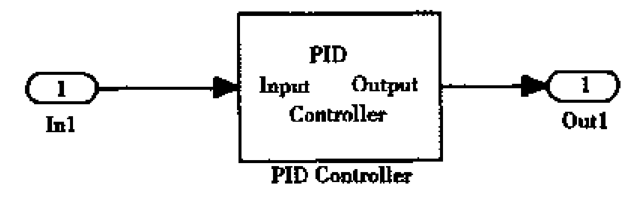


图 2－39　封装后的 PID 控制器模块

其次，在 Initialization 选项卡中设计封装后子系统模块的参数设置对话框，PID 控制器需要传递的变量参数有 4 个，单击 Add 按钮 4 次，即生成 4 个需要传递变量的位置。在 Prompt 文本框中输入 Proportional Kp 提示信息，在 Variable 文本框中输入变量参数 Kp，Control type 和 Assignment 采用默认值。按照这种方法依次设置 PID 控制器的其他 3 个参数。由于在生成封装子系统模块的过程中不需要传递变量，因此 Initialization commands 文本

框中不需要编写任何内容。设置完成后的 Initialization 选项卡如图 2－40 所示。

图 2－40 封装子系统模块参数设置对话框的设计

最后，在 Documentation 选项卡中，为 PID 控制器子系统模块添加文字说明。在 Block description 文本框中输入“this is a PID controller block. 这是一个 PID 控制器模块”；在 Block help 文本框中输入“this block can complete PID control and you can use it to simulate PID.”。完成各项参数设置之后，单击封装编辑器的 OK 按钮或 Apply 按钮，即完成了 PID 控制器子系统模块的整个封装过程。这时，双击 PID 控制器子系统模块则会弹出参数设置对话框，如图 2－41 所示。

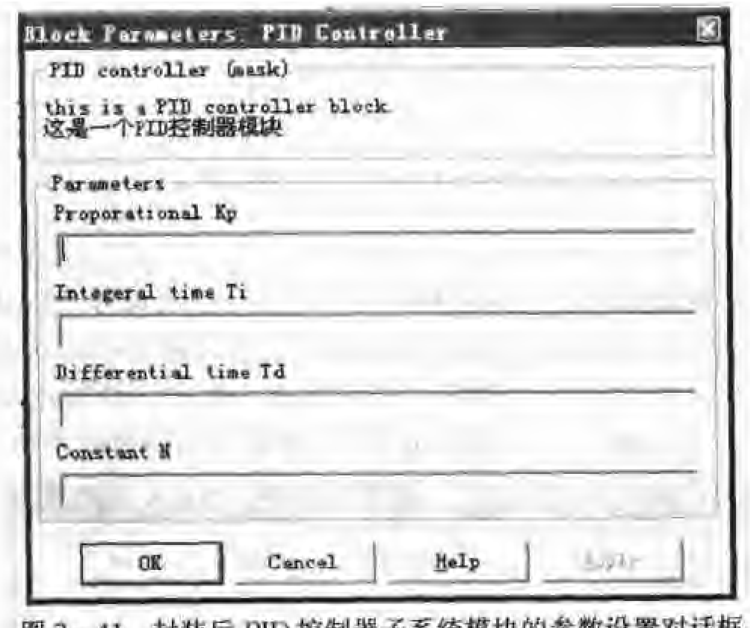


图 2－41 封装后 PID 控制器子系统模块的参数设置对话框

对封装后的子系统模块，可以在仿真模型窗口的菜单栏中执行 Edit/Edit mask 命令，在弹出的封装编辑器对话框中可以对子系统模块的封装参数进行修改。如果需要修改封装后子系统模块的内部结构，可以在选中子系统模块后，单击鼠标右键，在弹出的快捷菜

单中选择 Look under mask 命令，然后就可以在弹出的子系统模块仿真模型窗口中修改子系统模块的结构图。

2.5.3 条件执行子系统

在 SIMULINK 模块库中，Enable 模块和 Trigger 模块是比较特殊的模块，如果把这样的模块放到某个子系统中，则该子系统会在给定的控制信号的控制下执行，这样的子系统被称为条件执行子系统(conditionally executed subsystem)。

如果要创建的系统仿真模型比较复杂，而且它们的一些组件的执行受到另外一些组件的控制，则使用条件执行子系统是非常有用且必要的，尤其是在数字信号处理以及数字电路设计中，应用条件执行子系统进行建模将十分方便。

目前，SIMULINK 支持的条件执行子系统的结构、Enabled subsystem(使能子系统)、Triggered subsystem(触发子系统)、Enabled and triggered subsystem(触发使能子系统)等 3 种。

1. Enabled subsystem(使能子系统)

Enabled subsystem 只有在控制信号为正时，才执行子系统，否则将禁止执行。实际上它就是数字电路中的电平触发。当它处于禁止状态时，为了保持系统输出的连续性，它也有信号输出。用户可以选择继续保持禁止前的信号输出，也可以将子系统强制复位再指定输出值。

在 SIMULINK 4.1 版本中，Enabled subsystem 可以直接从 SIMULINK 的 Subsystems 模块库中获得。它在子系统模块中是独立的，不需要与其他模块进行连接，如图 2-42 所示。

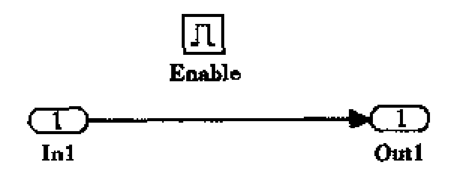


图 2-42 Enabled subsystem 模型

设置 Enable 模块参数的方法是首先双击 Enable 模块，然后在其模块参数设置对话框中，设置子系统重新使能的状态以及是否允许系统输出控制信号。Enable 模块参数设置对话框如图 2-43 所示。在 States when enabling 下列表框中可以选择 held 或 reset，同时，可以通过选定 Show output port 复选框以设置是否允许系统输出使能控制信号。这个功能对于 Enabled subsystem 中取决于包含在控制信号内的值的逻辑运算十分有用。

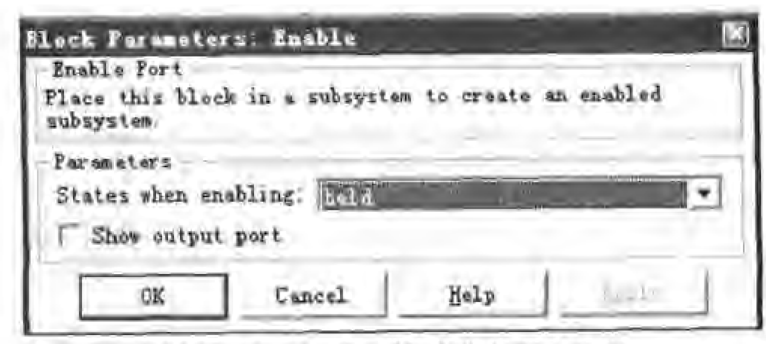


图 2-43 Enable 模块参数设置对话框

另外，在Enabled subsystem的Out1(输出)模块中，可以设置Enabled subsystem在禁止状态下的输出信号，设置方法是双击输出模块，这时将会弹出Block Parameters(禁止状态输出设置)对话框，如图2-44所示。在Output when disabled下拉列表框中，如果选择"held"，则在禁止状态下的输出将保持禁止前的状态值；如果选择"reset"，则子系统的输出将被强制复位，并且此时的输出需要在Initial output文本框中设置。



图2-44 禁止状态输出设置对话框

例2-7 *使用Enabled subsystem模块实现半波整流系统的仿真。*

半波整流系统在输入信号为正时，输出原信号，否则输出零。设计这个系统首先考虑的问题是如何判断输入信号是正(或负)，使用Enabled subsystem模块可以很容易地解决这个问题。半波整流系统的仿真模型如图2-45所示，其中的Enabled subsystem模型如图2-42所示。此例中需要把输出设置对话框中的Output when disabled设置为"held"，若设置为"reset"，则需要设置Initial output的值为零。

此半波整流系统的仿真结果如图2-46所示。

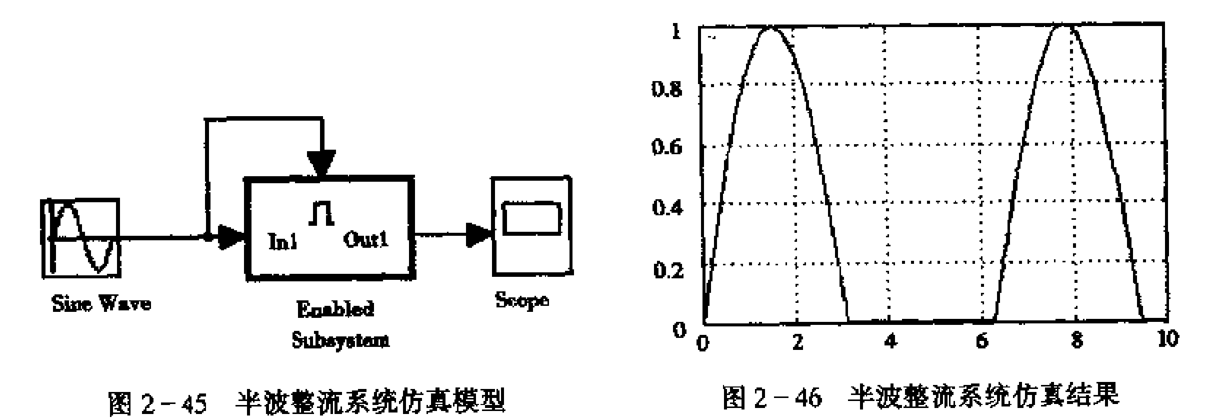


图2-45 半波整流系统仿真模型

图2-46 半波整流系统仿真结果

2. Triggered subsystem(触发子系统)

Triggered subsystem是在触发信号发生的瞬间执行子系统，然后保持子系统的输出状态，直到下一个触发信号到来。触发信号由触发输入信号状态决定，并且一个Triggered

subsystem 只有一个触发输入，它决定 Triggered subsystem 是否被执行。如同 Enabled subsystem 模块一样，Triggered subsystem 模块也可以直接从 SIMULINK 的 Subsystems 模块库中获得。

用户可以对触发信号进行如下的设置：

- 上升沿触发：当触发输入信号从零或负值变为正时，或者是从负值变为零时，则执行 Triggered subsystem。
- 下降沿触发：当触发输入信号从正值变为零或负值，或者是从零变为负值时，则执行 Triggered subsystem。
- 边沿触发：当输入信号发生变化时，即出现上升沿或下降沿时，均执行 Triggered subsystem。

另外，还有一种使 Triggered subsystem 执行的情况，就是函数调用触发，若采用这种触发方式，Triggered subsystem 的执行取决于 S 函数内部的逻辑，而与触发信号无关。

Triggered subsystem 模块如图 2－47 所示，与 Enabled subsystem 不同的是，Triggered subsystem 一旦执行就会将输出状态保持到下一个触发信号到来，因此不用设置状态是否保持以及禁止状态时子系统的输出。当 Triggered subsystem 需要设置触发类型（触发类型的四个选项前面已经介绍）时，可以根据实际需要进行设置。当用户在模块参数设置对话框中选定 Show output port 复选框时，则可以输出控制信号。这时需要用户选择输出信号的数据类型，用户可选择的数据类型包括 auto、double 和 int8，其中 auto 是默认选项，表示输出信号的数据类型与信号线连接的端口数据类型一致。

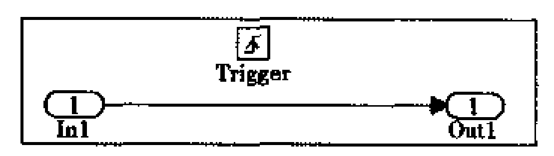


图 2－47　Triggered subsystem 模块

因为 Triggered subsystem 模块在仿真的特定时刻才被执行，因此，在 Triggered subsystem 中对使用的模块有一定的要求。例如要求模块能够继承采样时间（增益模块或逻辑运算模块等），或是采样时间设置为－1 的离散模块（这表示它的采样时间从驱动模块中继承）等。

例 2－8　*在这里以 Triggered subsystem 模块的触发信号为脉冲信号为例，说明 Triggered subsystem 模块的应用。*

例 2－8 仿真系统的仿真模型如图 2－48 所示。这里使用下降沿触发，仿真运行后的仿真结果如图 2－49 所示。

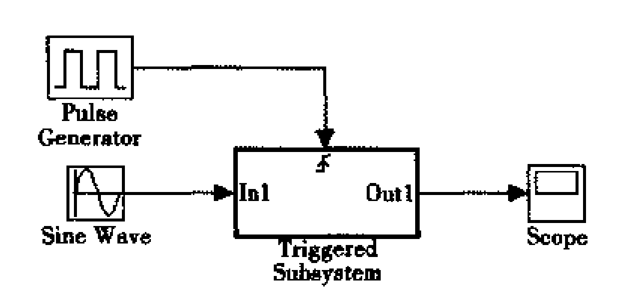


图 2－48　例 2－8 仿真模型

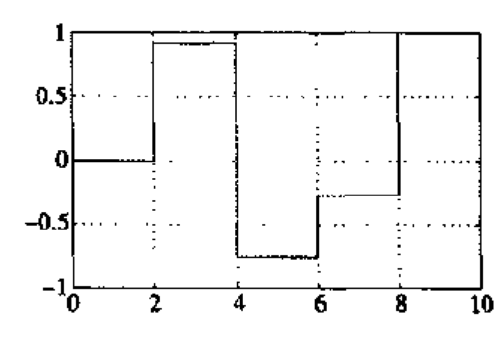


图 2－49　例 2－8 仿真结果

3.Enabled and triggered subsystem(触发使能子系统)

在 SIMULINK 的 Subsystems 模块库中还提供了 Enabled and triggered subsystem 模块,如图 2－50 所示。在触发使能子系统中,触发输入信号与使能触发信号是分开的,当触发信号发生时,SIMULINK 将判断使能输入信号是否为正,如果是则运行子系统,否则将禁止运行子系统。对 Enabled and triggered subsystem 模块这里不再做过多介绍,其参数的设置以及执行过程可以参考 Enabled subsystem 和 Triggered subsystem。

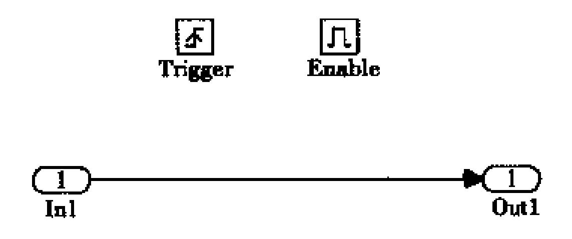


图 2－50　触发使能子系统模块

2.5.4　定义自己的模块库

如果用户建立了很多封装子系统模块,常常需要建立自己的模块库,用于分门别类地存储这些模块。另外,在进行仿真建模时,为了减少打开模块库的次数和方便系统仿真建模,用户通常也需要将实现某种功能的一组常用模块统一放置在同一模块库中。

创建自己的模块库的方法是:在 SIMULINK Library Browser 窗口的菜单栏中执行 File/New/Library 命令,将打开一个空白的模块库窗口,这时将需要存放在同一个模块库中的模块复制到模块库窗口中即可。创建这样一个模块库之后,在创建模型时就不需要再打开 SIMULINK 模块库,只需要在 MATLAB 命令窗口输入存放相应模块的模块库的文件名即可。

例如,为了方便模拟电路设计,可以创建一个文件名为“my _ ec”的模拟电路设计常用模块模块库,如图 2－51 所示。

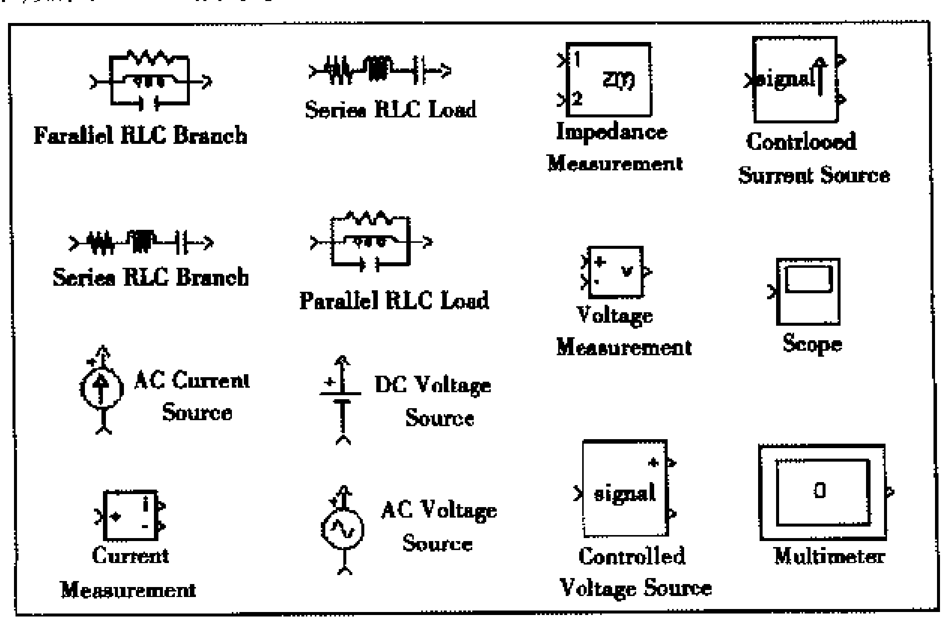


图 2－51　模拟电路设计常用模块模块库

创建了自己的模块库之后，还可以对该模块库的属性进行设置。其方法是：在所创建的模块库窗口的菜单栏中执行 File/Model properties 命令，这时将弹出 Model Properties（模块库属性）设置对话框，如图 2－52 所示，在该对话框中即可完成对模块库属性的设置。

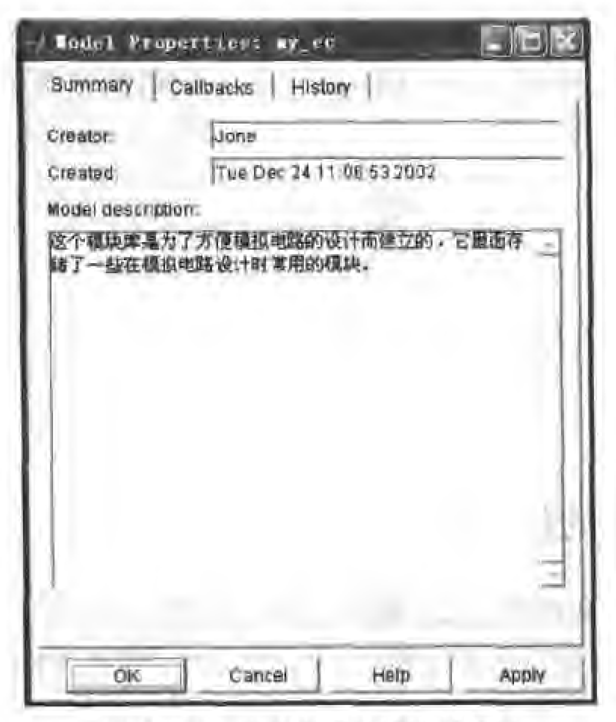


图 2－52　模块库属性设置对话框

2.6　动态仿真实例

通过前面的学习，读者已经掌握了在 SIMULINK 环境下创建系统模型、运行仿真以及进行仿真结果分析等的基本方法和技巧。在本节中将通过对“室内温度控制系统”进行仿真分析，介绍对系统进行动态仿真的实现过程。

“室内温度控制系统”的基本原理是：首先，给定保持室内的温度 70 华氏度，由于外部的温度是变化的（本例给出了室外温度的平均值是 50 华氏度，并且按正弦信号变化），因此需采用恒温器以及加热器来保证室内温度保持在 70 华氏度。同时，在温度调节过程中计算所花费的成本。需要注意的是，在进行温度控制时要把华氏温度转化为摄氏温度再进行控制，在进行结果输出时再还原为华氏温度。

1. 建立系统仿真模型

为了实现“室内温度控制系统”的功能，利用 SIMULINK 模块建立的系统仿真模型如图 2－53 所示。

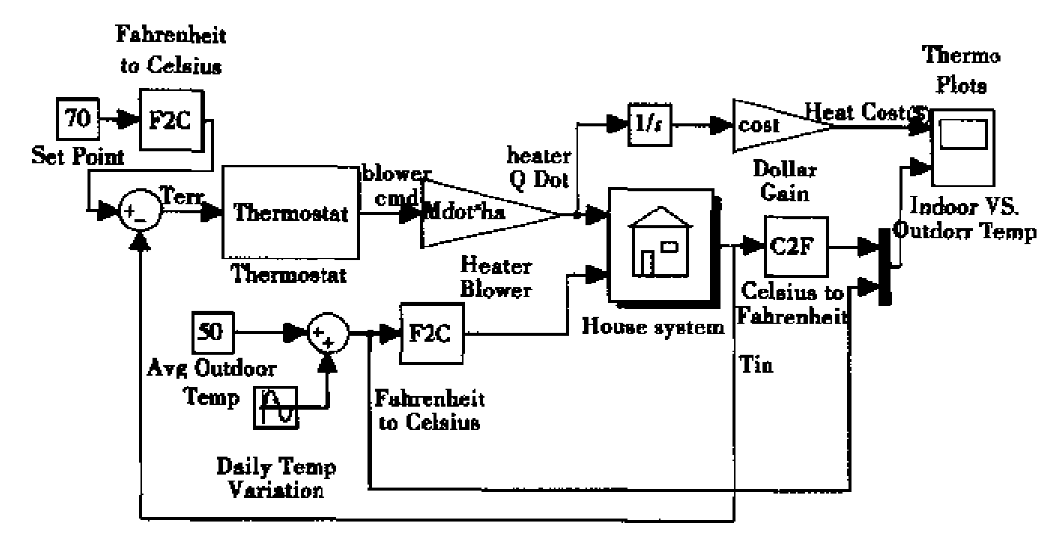


图 2－53　室内温度控制系统仿真模型

2．子系统的创建及参数设置

从图 2－53 可以看到，为了简化模型以及增加模型的可读性使用了子系统模块，并且进行了封装，如 Fahrenheit to Celsius 模块、Celsius to Fahrenheit 模块、Thermostat 模块以及 House system 模块都是经过封装的子系统模块。其中 Fahrenheit to Celsius 模块用于把华氏温度转化为摄氏温度，仿真模型如图 2－54(a)所示。Celsius to Fahrenheit 模块用于把摄氏温度转化为华氏温度，仿真模型如图 2－54(b)所示。双击这两个模块则可以分别打开如图 2－55 和 2－56 所示的模块参数对话框，从中可以看到温度转化的函数表达式。Thermostat 模块是恒温器封装子系统模块，它的功能是用非线性模块库中的继电器模块实现的，仿真模型如图 2－57 所示。House system 模块是房子子系统的仿真模型，它的图形标识是用 plot 绘图命令绘制的房子形状，为了引起读者的注意还为此模块增加了阴影显示。House system 子系统模块的内部结构如图 2－58 所示。

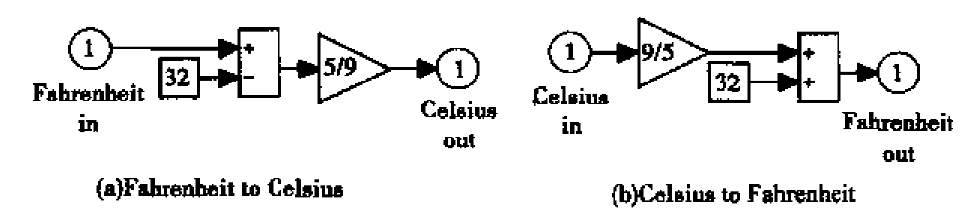


图 2－54　F2C 和 C2F 子系统模块

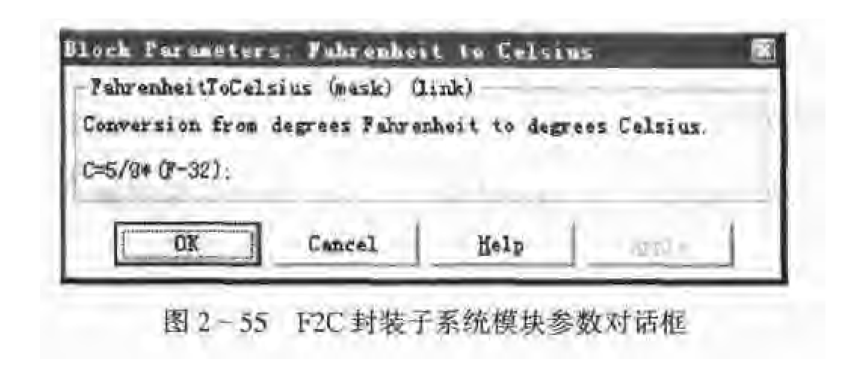


图 2－55　F2C 封装子系统模块参数对话框

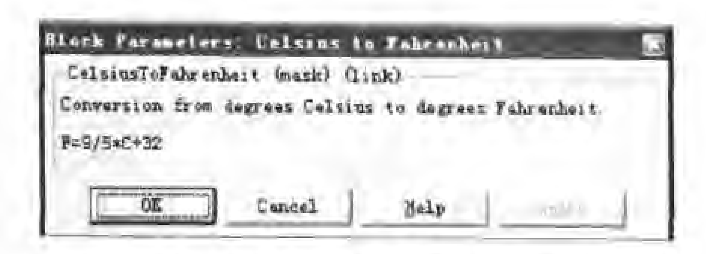


图 2－56　C2F 封装子系统模块参数对话框

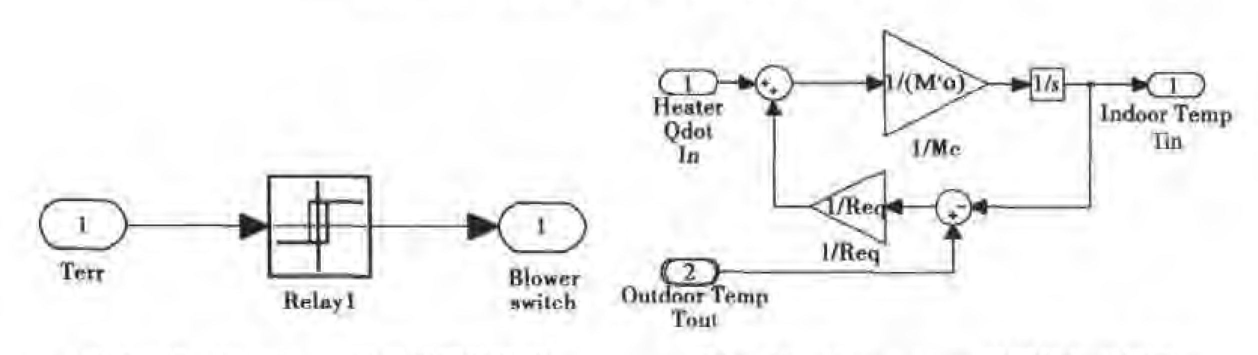


图 2－57　Thermostat 子系统模块模型　　　图 2－58　House system 子系统仿真模型

3. 运行仿真

建立了系统的仿真模型之后即可运行仿真，在本例中通过窗口菜单运行仿真。在运行仿真之前，需对仿真参数进行相应的设置，如图 2－59 所示。



图 2－59　系统仿真参数设置

4. 输出结果分析

本例采用输出基本模块——Scope 模块，进行系统仿真结果的输出与分析，仿真结果如图 2－60 所示。

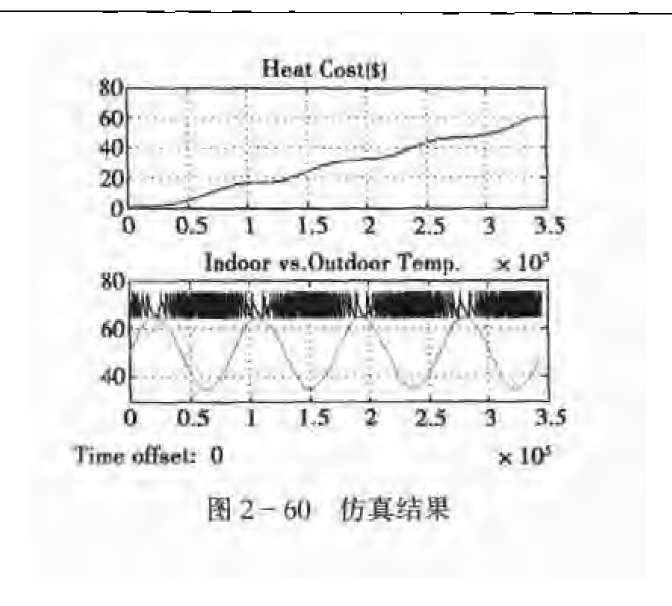


图2-60 仿真结果

2.7 本章小结

在本章中，首先系统地介绍了SIMULINK对系统仿真模块与信号线可以进行的基本操作，掌握了这些基本操作方法就可以熟练地创建系统仿真模型，特别是使用快捷键对系统仿真模块与信号线进行操作，有时候会达到事半功倍的效果。

其次介绍了仿真模型的调试和运行方法。仿真模型的调试既可以通过图形界面调试器实现，也可以通过MATLAB命令实现。在运行仿真模型之前需要对各项仿真参数进行设置，当然也可以采用默认参数运行仿真。仿真运行的方法包括使用命令运行和使用窗口菜单运行两种。进行系统仿真的最关键问题就是对仿真结果进行分析。常用的仿真结果分析方法是利用输出模块对仿真结果进行分析，这种方法不仅简单而且直观。有时候为了简化系统仿真模型，增加其可读性，往往需要创建子系统模块，本章系统地介绍了子系统模块的创建方法及其应用，能够使读者对其有一个较为全面地了解。

通过本章的学习，读者应该能够对SIMULINK仿真工具有一个全面的认识和了解，能够熟练地掌握运用SIMULINK进行系统的建模以及仿真和分析的方法和技巧，能够正确地运用仿真调试器对系统模型进行调试，能够创建、封装子系统模块以及掌握条件子系统模块的特性，为学习后续的知识打下良好的基础。

习 题

1. 利用所了解的方法对Transfer Fun模块进行创建、复制、旋转、改变大小以及增加阴影等模块基本操作，并把它的模块参数denominator设置为[2 3 1]。
2. 利用SIMULINK中的基本模块建立系统开环传递函数：

$$G(S)=\frac{(2s+1)(s^2+3s+1)}{(s^2+0.1s+3)(0.1s+5)(3s+2)}$$

如果系统具有单位负反馈，试求出系统的闭环传递函数。

3. 在练习题 2 中，如果输入信号是阶跃信号，利用菜单和命令两种方式运行仿真，并使用示波器输出模块观察系统的仿真结果。

4. 利用 SIMULINK 模块库中的基本模块建立如图 2－61 所示的仿真模型。

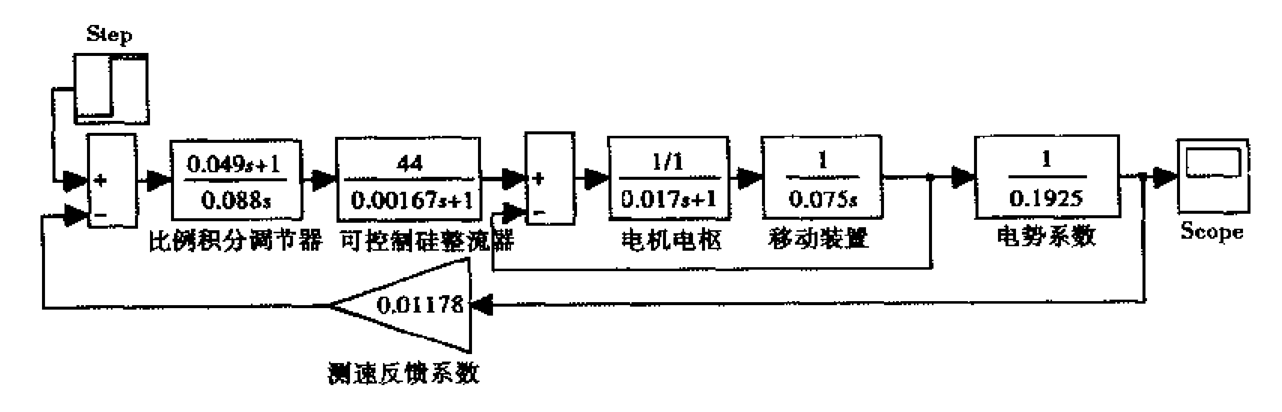


图 2－61　仿真模型

5. 创建如图 2－62 所示的锅炉设备模型的 SIMULINK 仿真模型，然后把它组建为子系统模块并对其进行封装。

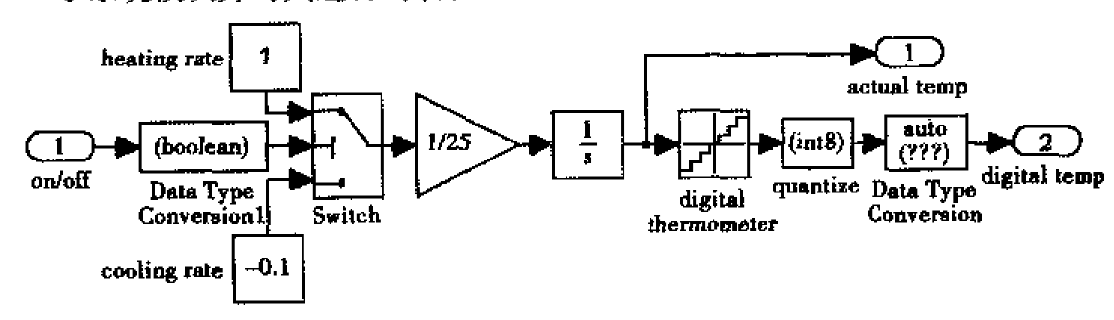


图 2－62　锅炉设备仿真模型

6. 对练习题 4 和 5 中的系统进行调试，使用单步执行系统，观察系统的仿真过程，并求出练习题 4 中所创建的系统的平衡点。

第 3 章　SIMULINK 动态仿真扩展

知识点：

- 用语句修改 SIMULINK 模型
- S 函数
- M 文件 S 函数
- C/C+ + S 函数
- Stateflow 原理与应用
- 基于 SIMULINK 的 VR 技术

在第 2 章中已经系统地讲解了利用 SIMULINK 进行系统建模、仿真以及仿真结果分析的基本方法和技巧。在本章中将介绍利用 SIMULINK 进行系统动态仿真的高级应用。首先介绍如何利用 MATLAB 提供的函数和命令绘制系统仿真模型，然后介绍实现特定功能的仿真模块的创建，即 S 函数模块的创建。在这里主要介绍利用 MATLAB 编程语言、C/C+ +语言编写 S 函数的方法与技巧。最后介绍 Stateflow 的原理与应用，以及输出结果的可视化方法——虚拟现实（VR）技术。通过本章的学习，读者可以更加深入地理解 MATLAB 中的 SIMULINK 动态仿真环境。

3.1　用语句修改 SIMULINK 模型

利用第 2 章介绍的方法对仿真模型进行操作，如创建、调试、运行以及仿真结果的分析等都非常简单方便，但在实现某些功能的时候，用户可能希望能够通过 MATLAB 语句与命令实现这些操作。本节将详细介绍如何利用 MATLAB 语句与命令对 SIMULINK 模型进行操作。

3.1.1　修改 SIMULINK 模型的语句与命令

使用 MATLAB 语句或命令创建、修改 SIMULINK 模型时，常用的命令如表 3 - 1 所示。

表 3-1 创建、修改模型的命令列表

命令	功能说明
ADD_BLOCK	向仿真模型中加入一个新模块
ADD_LINE	向仿真模型中加入一条新信号线
BDCLOSE	关闭一个仿真模型窗口
BDROOT	获取根仿真模型名称
CLOSE_SYSTEM	关闭一个仿真模型
DELETE_BLOCK	从仿真模型中删除模块
DELETE_LINE	从仿真模型中删除信号线
FIND_SYSTEM	查找模型、模块、信号线或注释
GCB	获取当前模块的路径名
GCBH	获取当前模块的句柄
GCS	获取当前仿真模型的路径名
GET_PARAM	获取仿真模型或模块的参数值
NEW_SYSTEM	创建一个新的仿真模型
OPEN_SYSTEM	打开一个已经存在的仿真模型
REPLACE_BLOCK	替换仿真模型中的模块
SAVE_SYSTEM	保存仿真模型
SET_PARAM	设置仿真模型或模块的参数值
SIMULINK	打开仿真模块库

在创建、修改 SIMULINK 模型时常用的命令如下：

add_block 命令

add_block 命令的作用是向 SIMULINK 仿真模型中增加一个新模块。其调用格式为：

```
add_block('sname','dname','parameter1',value1,…)
```

其中，sname 是源模块名，dname 是目标模块名，parameter 和 value 分别是模块参数的属性名和属性值。这个命令是把一个含有完整路径的模块'sname'添加到指定目标模块名和路径的'dname'模型窗口中，并且按所列出来的参数属性值修改源模块，如果省略了属性名和属性值参数选项，则目标模块参数与源模块参数一致。

例 3-1 从 SIMULINK 模块库的 Continous 子模块库中把 Integrator 模块添加到当前路径下模型名为"li3_1"的模型中，并把模块名称改为 Integrator_fun。

命令的具体调用格式如下：

```
>> add_block('built-in/Integrator','li3_1/Integrator_fun');
```

add_line 命令

add_line 命令的作用是向 SIMULINK 仿真模型中添加信号线。其调用格式为：

```
add_line('sysname', 'outport', 'inport')
```

或

```
add_line('sysname',points)
```

其中，命令 add_line('sysname', 'outport', 'inport')的作用是在指定的 sysname 仿真模型中，从指定的模块的输出端口'outport'到指定的模块的输入端口'inport'添加一条信号线。命令 add_line('sysname',points)的作用是增加一条线段到指定的仿真模型中，数组 points 的每一行都用来指定线段端点的 X、Y 坐标。

delete _ line 命令

delete _ line 命令的作用是从 SIMULINK 仿真模型中删除信号线。其具体用法和调用格式与前面讲到的 add _ line 命令相同,这里就不再介绍。

delete _ block 命令

delete _ block 命令的作用是从 SIMULINK 仿真模型中删除模块。其具体调用格式为:

```
delete _ block('name')
```

命令 delete _ block('name')的作用是从仿真模型中删除一个包含完整路径名的模块名为 name 的模块。

replace _ block 命令

replace _ block 命令的作用是在 SIMULINK 仿真模型中替换模块。其具体调用格式为:

```
replace _ block('sysname', 'block1', 'block2', 'np')
```

或

```
replace _ block('sysname', 'parameter1',value1, 'bname1',…)
```

其中,命令 replace _ block('sysname', 'block1', 'block2', 'np')的作用是用'block2'替换'sysname'仿真模型中所有类型为'block1'的模块;'np'参数选项可以省略,如果命令中不包含此参数选项,则在替换模块之前会弹出一个要求选择匹配的模块的提示对话框,如果有这个参数选项,则不弹出提示对话框。命令 replace _ block('sysname', 'parameter1', value1, 'bname1',…)的作用是用模块名为'bname'的模块替换'sysname'仿真模型中'所有指定的参数属性具有指定值的模块。

另外,进行模型模块参数的获取或是设置的命令 get _ param 和 set _ param 前面已经介绍过,具体用法和调用格式请参见 2.3 节中的介绍。一些对 SIMULINK 文件进行操作的语句与命令,完全可以由仿真窗口的 File 菜单中的子菜单实现。常用的命令如下:

new _ system 命令

new _ system 命令用于创建一个新的 SIMULINK 仿真模型。它的具体调用格式为:

```
new _ system('sysname')
```

命令 new _ system('sysname')表示创建一个包含完整路径的模型名为'sysname'的新仿真模型。

open _ system 命令

open _ system 命令的作用是打开一个 SIMULINK 仿真模型窗口或是仿真模块的对话框。它的具体调用格式是:

```
open _ system('system')
```

或

```
open _ system('bname')
```

其中,命令 open _ system('system')用于打开一个指定的包含完整路径名的'sysname' SIMULINK 模型。命令 open _ system('bname')用于打开一个与指定的包含完整路径名的'bname'模块相关的对话框。

另外,open _ system 命令在调用时如果再增加一个参数选项'force',此时命令的作用

则是查看指定的'bname'封装子系统模块，相当于右键菜单中 Look under mask 命令的作用。它的调用格式如下：

open _ system('bname', 'force')

save _ system 命令

save _ system 命令的作用是保存 SIMULINK 仿真模型。它的调用格式是：

save _ system('sysname', 'newname')

命令 save _ system('sysname', 'newname')表示用指定的包含完整路径的'newname'保存指定的'sysname'仿真模型文件，此仿真模型必须是打开的。此外，当 save _ system 命令不带任何参数选项时，即命令 save _ system，表示用当前的文件名保存当前的 SIMULINK 仿真模型。当 save _ system 命令只带有'sysname'参数时，表示用当前的文件名将当前的 SIMULINK 仿真模型保存到指定的'sysname'仿真模型文件中。注意，此时这个仿真模型窗口必须是打开的。

close _ system 命令

close _ system 命令的功能是关闭 SIMULINK 仿真模型窗口或是模块的对话框。它的调用格式是：

close _ system

close _ system('sysname')

close _ system('sysname',saveflag)

close _ system('sysname', 'newname')

close _ system('bname')

当 close _ system 命令不带任何参数选项时，则表示关闭当前的 SIMULINK 仿真模型窗口，同时它会询问是否保存修改。当该命令只带有'sysname'参数选项时，表示关闭指定的'sysname'仿真模型窗口。当该命令同时带有'sysname'和'saveflag'参数选项时，表示关闭指定的'sysname'仿真模型窗口，如果 saveflag=0，系统将不保存对仿真模型的修改就关闭仿真模型窗口。如果 saveflag=1，系统将以当前的名称保存仿真模型后关闭仿真模型窗口。当该命令同时带有'sysname'和'newname'参数选项时，系统将以'newname'保存指定的'sysname'仿真模型窗口。当该命令带有'bname'参数时，将关闭与'bname'模块相关的对话框。

bdclose 命令

bdclose 命令的作用是无条件关闭所有 SIMULINK 窗口。它的调用格式是：

bdclose('sysname')

或

bdclose('all')

命令 bdclose('sysname')的作用是关闭指定的'sysname'仿真模型窗口，而命令 bdclose('all')则表示关闭所有仿真模型窗口。此外，这个命令还可以不带任何参数选项，这时它的作用是无条件关闭当前的仿真模型窗口，将不保存从前一次保存到关闭之前所作的任何修改。

3.1.2　SIMULINK 模型文件与模块属性

SIMULINK 提供了框图和 MATLAB 命令两种方式，用于系统仿真模型的创建、设

置和模型模块参数的修改。同时,SIMULINK 还允许用户直接修改仿真模型文件。下面介绍 SIMULINK 模型文件。

SIMULINK 把每一个仿真模型保存在以 .mdl 为扩展名的文件中,用户可以通过 MATLAB 文件编辑窗口打开模型文件。需要提醒读者的是,在打开仿真模型文件之前,需要把打开文件的类型设置成 MATLAB Files 或 All Files,否则将看不到需要打开的扩展名为 .mdl 的模型文件。如图 3-1 所示是打开图 2-31 所示的系统仿真模型的模型文件。



图 3-1 仿真模型文件

模型文件的内容包含了模型模块的属性名称和属性值,下面以图 2-31 所示的系统仿真模型文件为例进行介绍。它的模型文件的结构及内容如下:

```
Model {                              %模型模块引导语句
  Name                    "li2_14"
  Version                 4.00
  SampleTimeColors        off
  LibraryLinkDisplay      "none"
  WideLines               off
  ShowLineDimensions      off
  ShowPortDataTypes       off
  ShowStorageClass        off
  ExecutionOrder          off
  RecordCoverage          off
  CovPath                 "/"
  CovSaveName             "covdata"
  CovMetricSettings       "dw"
  CovNameIncrementing     off
  CovHtmlReporting        on
  BlockNameDataTip        off
  BlockParametersDataTip  off
  BlockDescriptionStringDataTip  off
  ToolBar                 on
  StatusBar               on
```

```
BrowserShowLibraryLinks off
BrowserLookUnderMasks   off
Created         "Wed Dec 25 14:41:04 2002"
UpdateHistory           "UpdateHistoryNever"
ModifiedByFormat        "%<Auto>"
LastModifiedBy    "lina"
ModifiedDateFormat      "%<Auto>"
LastModifiedDate        "Wed Jan 01 21:54:42 2003"
ModelVersionFormat      "1.%<AutoIncrement:5>"
ConfigurationManager    "None"
SimParamPage            "Solver"
StartTime         "0.0"
StopTime          "10.0"
SolverMode              "Auto"
Solver            "ode45"
RelTol            "1e-3"
AbsTol            "auto"
Refine            "1"
MaxStep           "auto"
MinStep           "auto"
MaxNumMinSteps     "-1"
InitialStep             "auto"
FixedStep         "auto"
MaxOrder          5
OutputOption            "RefineOutputTimes"
OutputTimes             "[]"
LoadExternalInput       off
ExternalInput           "[t, u]"
SaveTime          on
TimeSaveName            "tout"
SaveState         off
StateSaveName           "xout"
SaveOutput              on
OutputSaveName    "yout"
LoadInitialState        off
InitialState            "xInitial"
SaveFinalState    off
FinalStateName    "xFinal"
SaveFormat              "Array"
LimitDataPoints   on
MaxDataPoints           "1000"
Decimation              "1"
AlgebraicLoopMsg        "warning"
```

```
MinStepSizeMsg    "warning"
UnconnectedInputMsg    "warning"
UnconnectedOutputMsg    "warning"
UnconnectedLineMsg    "warning"
InheritedTsInSrcMsg    "warning"
SingleTaskRateTransMsg  "none"
MultiTaskRateTransMsg  "error"
IntegerOverflowMsg    "warning"
CheckForMatrixSingularity "none"
UnnecessaryDatatypeConvMsg "none"
Int32ToFloatConvMsg    "warning"
InvalidFcnCallConnMsg  "error"
SignalLabelMismatchMsg  "none"
LinearizationMsg    "none"
VectorMatrixConversionMsg "none"
SfunCompatibilityCheckMsg "none"
BlockPriorityViolationMsg "warning"
ArrayBoundsChecking    "none"
ConsistencyChecking    "none"
ZeroCross    on
Profile    off
SimulationMode    "normal"
RTWSystemTargetFile    "grt.tlc"
RTWInlineParameters    off
RTWRetainRTWFile    off
RTWTemplateMakefile    "grt_default_tmf"
RTWMakeCommand    "make_rtw"
RTWGenerateCodeOnly    off
TLCProfiler    off
TLCDebug    off
TLCCoverage    off
AccelSystemTargetFile  "accel.tlc"
AccelTemplateMakefile  "accel_default_tmf"
AccelMakeCommand    "make_rtw"
TryForcingSFcnDF    off
ExtModeMexFile    "ext_comm"
ExtModeBatchMode    off
ExtModeTrigType    "manual"
ExtModeTrigMode    "normal"
ExtModeTrigPort    "1"
ExtModeTrigElement    "any"
ExtModeTrigDuration    1000
ExtModeTrigHoldOff    0
```

```
ExtModeTrigDelay          0
ExtModeTrigDirection      "rising"
ExtModeTrigLevel          0
ExtModeArchiveMode        "off"
ExtModeAutoIncOneShot     off
ExtModeIncDirWhenArm      off
ExtModeAddSuffixToVar     off
ExtModeWriteAllDataToWs off
ExtModeArmWhenConnect     on
ExtModeSkipDownloadWhenConnect off
ExtModeLogAll             on
ExtModeAutoUpdateStatusClock on
OptimizeBlockIOStorage    on
BufferReuse               on
ParameterPooling          on
BlockReductionOpt         on
RTWExpressionDepthLimit 5
BooleanDataType    off
BlockDefaults {                                    %默认的模块设置
  Orientation             "right"
  ForegroundColor         "black"
  BackgroundColor         "white"
  DropShadow              off
  NamePlacement       "normal"
  FontName                "Helvetica"
  FontSize                10
  FontWeight              "normal"
  FontAngle               "normal"
  ShowName                on
}
AnnotationDefaults {                               %默认的模块标注设置
  HorizontalAlignment     "center"
  VerticalAlignment       "middle"
  ForegroundColor         "black"
  BackgroundColor         "white"
  DropShadow              off
  FontName                "Helvetica"
  FontSize                10
  FontWeight              "normal"
  FontAngle               "normal"
}
LineDefaults {                                     %默认的信号线设置
  FontName                "Helvetica"
```

```
    FontSize                9
    FontWeight              "normal"
    FontAngle               "normal"
  }
  System {                                        %默认的系统参数设置
    Name                    "li2_14"
    Location                [480, 93, 980, 386]
    Open                    on
    ModelBrowserVisibility  off
    ModelBrowserWidth       200
    ScreenColor             "automatic"
    PaperOrientation        "landscape"
    PaperPositionMode       "auto"
    PaperType               "A4"
    PaperUnits              "centimeters"
    ZoomFactor              "100"
    ReportName              "simulink-default.rpt"
    Block {
      BlockType             Integrator
      Name                  "Integrator"
      Ports                 [1, 1]
      Position              [175, 95, 205, 125]
      ExternalReset         "none"
      InitialConditionSource "internal"
      InitialCondition      "0"
      LimitOutput           off
      UpperSaturationLimit  "inf"
      LowerSaturationLimit  "-inf"
      ShowSaturationPort    off
      ShowStatePort         off
      AbsoluteTolerance     "auto"
    }
    Block {
      BlockType             Step
      Name                  "Step"
      Position              [45, 95, 75, 125]
      Time                  "0"
      Before                "0"
      After                 "1"
      SampleTime            "0"
      VectorParams1D        on
    }
    Block {
```

```
    BlockType               Sum
    Name                    "Sum"
    Ports                   [2, 1]
    Position                [125, 100, 145, 120]
    ShowName                off
    IconShape               "round"
    Inputs                  "|+-"
    SaturateOnIntegerOverflow   on
  }
  Block {
    BlockType               ToWorkspace
    Name                    "To Workspace"
    Position                [360, 95, 420, 125]
    VariableName            "simout"
    MaxDataPoints           "inf"
    Decimation              "1"
    SampleTime              "-1"
    SaveFormat              "Structure"
  }
  Block {
    BlockType               TransferFcn
    Name                    "Transfer Fcn"
    Position                [245, 92, 305, 128]
    Numerator               "[1]"
    Denominator             "[1 1]"
    AbsoluteTolerance       "auto"
    Realization             "auto"
  }
  Line {
    SrcBlock                "Step"
    SrcPort                 1
    DstBlock                "Sum"
    DstPort                 1
  }
  Line {
    SrcBlock                "Sum"
    SrcPort                 1
    DstBlock                "Integrator"
    DstPort                 1
  }
  Line {
    SrcBlock                "Integrator"
    SrcPort                 1
```

```
      DstBlock              "Transfer Fcn"
      DstPort               1
    }
    Line {
      SrcBlock              "Transfer Fcn"
      SrcPort               1
      Points                [15, 0]
      Branch {
DstBlock              "To Workspace"
DstPort               1
      }
      Branch {
Points                [0, 55; -190, 0]
DstBlock              "Sum"
DstPort               2
      }
    }
  }
}
```

从中可以看出，SIMULINK 的模型文件包含了系统模型的所有属性，并且由以下五个不同的部分组成：

- 模型参数的 Model{}部分，这部分主要包括模型名称、SIMULINK 版本序号及仿真参数等可以通过仿真参数对话框设置的所有仿真参数。
- 模块参数默认设置的 BlockDefaults{}部分。
- 模型中模块标注默认设置的 AnnotationDefaults{}部分。
- 模型中信号线默认设置的 LineDefaults{}部分。
- 描述整个仿真模型系统以及各个子系统的 System{}部分，这部分主要包含了模型本身以及各个模块、信号线和标注等的说明信息。

3.1.3　用语句实现仿真模型的绘制

下面以图 2－31 所示的系统仿真模型为例，介绍使用 MATLAB 语句和命令实现模型的绘制的方法与技巧。

例 3－2　使用 MATLAB 语句和命令建立如图 2－31 所示的仿真模型。

首先使用命令 new_system()和 open_system()创建并打开一个新的仿真模型窗口，新模型的名称为"li3_1"；然后使用 MATLAB 语句和命令添加仿真模型所需要的模块、信号线等，在进行参数修改和设置时可以使用 set_param()命令来实现；最后对所创建的仿真模型进行保存和关闭。创建该仿真模型的 MATLAB 语句和命令如下：

```
% MATLAB command 3-2
>> new_system('li3_1');
>> open_system('li3_1');
```

```
>> add_block('built-in/Step','li3_1/Input signal');
>> add_block('built-in/Sum','li3_1/sum');
>> add_block('built-in/Integrator','li3_1/Integrator');
>> add_block('built-in/Transfer Fcn','li3_1/Transfer Fcn');
>> add_block('built-in/To Workspace','li3_1/To Workspace');
>> add_line('li3_1','Input signal/1','sum/1');
>> add_line('li3_1','sum/1','Integrator/1');
>> add_line('li3_1','Integrator/1','Transfer Fcn/1');
>> add_line('li3_1','Transfer Fcn/1','To Workspace/1');
>> set_param('li3_1','Location',[480 93 980 386]);
>> set_param('li3_1/Integrator','Position',[175 95 205 125]);
>> set_param('li3_1/Integrator','Position',[175 95 205 125]);
>> set_param('li3_1/Input signal','Position',[45 95 75 125]);
>> set_param('li3_1/sum','Position',[125 100 145 120]);
>> set_param('li3_1/sum','IconShape','round');
>> set_param('li3_1/sum','Inputs','|+-');
>> set_param('li3_1/To Workspace','Position',[360 95 420 125]);
>> set_param('li3_1/To Workspace','VariableName','simout');
>> set_param('li3_1/Transfer Fcn','Position',[245 92 305 128]);
>> set_param('li3_1/Transfer Fcn','Denominator','[1 1]');
>> add_line('li3_1',[340 110; 340 150; 135 150; 135 120]);
>> save_system('li3_1');
>> close_system('li3_1');
```

使用上面的 MATLAB 语句和命令创建的仿真模型和图 2－31 所示的仿真模型是相同的，如图 3－2 所示。

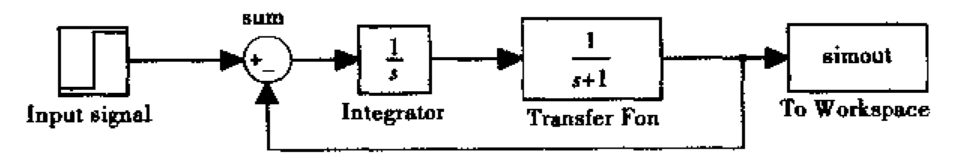


图 3－2　通过 MATLAB 语句和命令创建的仿真模型

3.2　S 函数

通过前面的介绍，读者对使用 SIMULINK 进行仿真系统的建模方法已经有所了解，对于绝大多数的系统模型都可以通过 SIMULINK 提供的模块库进行创建，但是在创建一些复杂的或具有特殊功能的仿真系统时，用户就需要自己创建相应的模块。这时，用户可以通过编写 S 函数（即 System Function）的方法来实现，这是对 SIMULINK 功能的扩展，也是它的一个显著特点。掌握了 S 函数的编写方法，可以充分发挥 SIMULINK 的功能，否则就只能利用 SIMULINK 模块库中的基本模块或是子系统模块来搭建仿真模型，而在某些时候它却不能满足用户的需要。接下来将详细介绍 S 函数的编写方法。

3.2.1　S 函数简介

S 函数的实质是使用特殊调用方法的 MATLAB 函数，它利用计算机程序语言来描述动态系统。在 SIMULINK 4.0 版本中，允许使用 MATLAB 程序语言、C/C++语言以及 Fortran 语言等进行 S 函数的编写。S 函数的功能十分强大，它不仅支持连续系统和离散系统，还支持混合系统。因此，绝大部分 SIMULINK 仿真模型均可以由 S 函数进行描述。

虽然 S 函数的编写十分简单方便，但是它却可以实现非常复杂的功能。它的编写步骤如下：

(1)编写 S 函数程序。程序的编写有一定的模式，SIMULINK 为用户提供了编写 S 函数的模板文件，只需要在必要的子函数内编写相应的程序代码并输入相应的参数即可。

(2)从 SIMULINK 的 Function & Table 模块库中复制一个 S 函数模块，然后在模块参数设置对话框中设置相应的参数即可。

在 S 函数程序编写完成之后，可以通过 SIMULINK 的 Function & Table 子模块库中的 S 函数模块把 S 函数添加到 SIMULINK 仿真模型中，并可以通过 S 函数模块参数设置对话框对指定的 S 函数的名称以及需要传递的附加参数进行设置。S 函数的模块仿真参数对话框如图 3-3 所示。

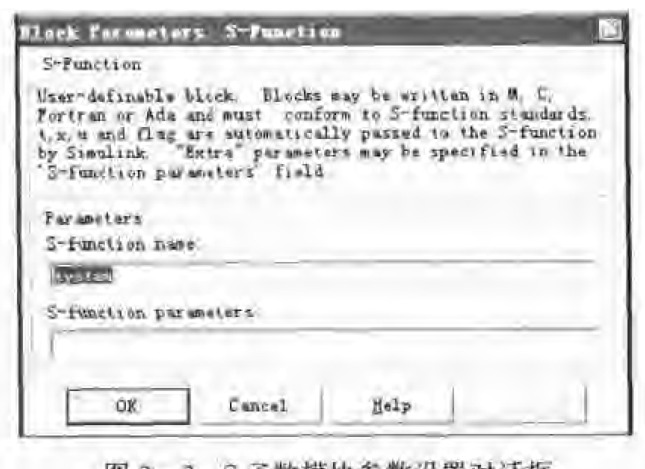


图 3-3　S 函数模块参数设置对话框

另外，通过对 S 函数模块进行封装，使得设置 S 函数需要传递的仿真参数变得更加方便、快捷。S 函数封装的方法和前面介绍的对子系统模块进行封装的方法是一样的，这里就不再赘述。

3.2.2　S 函数原理

S 函数的调用与 SIMULINK 模块的仿真系统执行过程是对应的，在不同的仿真阶段调用与之对应的子函数。SIMULINK 模块的仿真执行流程如图 3-4 所示。

从图 3-4 可以看出，在仿真过程开始时，SIMULINK 首先将模型进行初始化，然后进入仿真循环，在仿真循环的各个阶段，SIMULINK 都会调用仿真模块，完成一个仿真循

环后，就进入下一个仿真时间步，依次循环下去直到仿真结束。调用模型中 S 函数仿真模块的过程与 SIMULINK 模块仿真执行流程是一样的。对具体仿真执行流程说明如下：

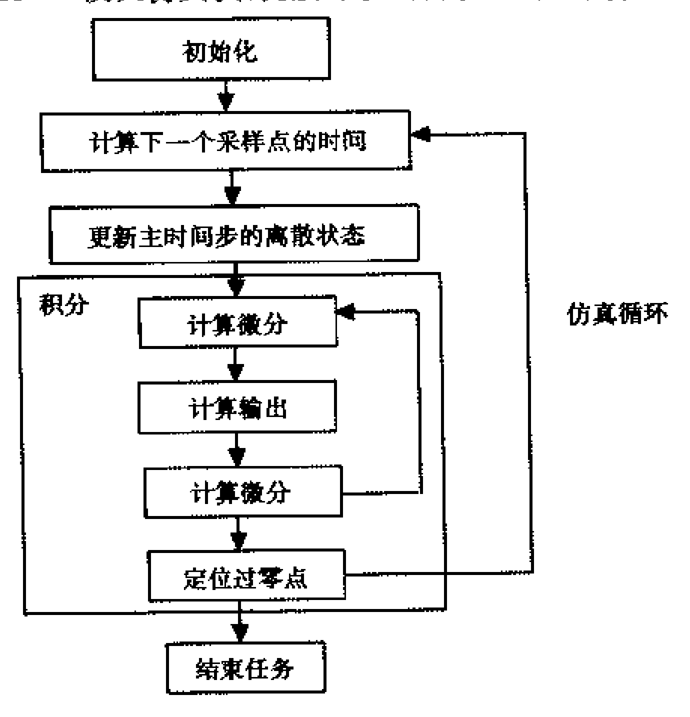


图 3－4　仿真执行流程图

1）初始化

初始化是进行仿真的第一阶段，这个阶段中将初始化 S 函数的如下信息：

- 初始化 Simstruct（包含 S 函数信息的数据结构）。
- 确定输入、输出端口的数目大小。
- 确定采样时间。
- 分配内存和 sizes 数组。

2）计算下一个采样时间

如果采用的是变步长的求解解法，这一步是必须的，它将确定下一个仿真时间步的大小。

3）计算主仿真时间步的输出

只有计算出了当前主时间步的输出，计算主仿真时间步的输出才能作为其他模块的有效输入，才能对其他模块有作用。

4）更新主时间步的离散状态

在这个阶段中，S 函数需要进行当前时间的仿真循环的离散状态更新。

5）积分

在积分阶段，SIMULINK 将按最小时间步来调用 S 函数的输出和对 S 函数进行微分，若 S 函数具有非采样过零点（只有 C 语言的 S 函数才有可能存在这种情况），则 SIMULINK也会按最小时间步来调用 S 函数的输出与过零点，以达到确定过零点位置的目的。

6)结束任务

在结束任务阶段,用户可以指定仿真完成某些特殊的操作。

3.2.3 S函数实例

接下来,通过一个S函数实例,介绍S函数的编写和子函数的调用格式,读者可以不必理解每条语句的具体含义。

例3-3 利用S函数实现将输入信号扩大2倍,再进行输出的功能。

在该实例中,使用由MATLAB编程语言编写的S函数timestwo.m。程序代码如下:

```
% MATLAB program 3-3
function [sys,x0,str,ts] = timestwo(t,x,u,flag)
%TIMESTWO S-function whose output is two times its input.
%   This M-file illustrates how to construct an M-file S-function that
%   computes an output value based upon its input. The output of this
%   S-function is two times the input value:
%     y = 2 * u;
%   See sfuntmpl.m for a general S-function template.
%   See also SFUNTMPL.
%   Copyright 1990-2001 The MathWorks, Inc.
%    $Revision: 1.6 $
% Dispatch the flag.The switch function controls the calls to
% S-function routines at each simulation stage of the S-function.
switch flag,
  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  % Initialize the states, sample times, and state ordering strings.
  case 0
    [sys,x0,str,ts]=mdlInitializeSizes;
  %%%%%%%%%%%
  % Outputs %
  %%%%%%%%%%%
  % Return the outputs of the S-function block.
  case 3
    sys=mdlOutputs(t,x,u);
  %%%%%%%%%%%%%%%%%%%
  % Unhandled flags %
  %%%%%%%%%%%%%%%%%%%
  % There are no termination tasks (flag=9) to be handled.
  % Also, there are no continuous or discrete states,
  % so flags 1,2, and 4 are not used, so return an emptyu
  % matrix
```

```
  case {1, 2, 4, 9}
    sys=[];
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Unexpected flags (error handling)%
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Return an error message for unhandled flag values.
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
% end timestwo
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=============================================================================
function [sys,x0,str,ts] = mdlInitializeSizes()
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = -1;  % dynamically sized
sizes.NumInputs      = -1;  % dynamically sized
sizes.DirFeedthrough = 1;   % has direct feedthrough
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
str = [];
x0  = [];
ts  = [-1 0];  % inherited sample time
% end mdlInitializeSizes
%=============================================================================
% mdlOutputs
% Return the output vector for the S-function
%=============================================================================
function sys = mdlOutputs(t,x,u)
sys = u * 2;
% end mdlOutputs
```

3.3　M文件S函数

用MATLAB编程语言编写的S函数通常称之为M文件S函数。在M文件S函数中,对S函数子函数的调用是通过M文件子函数实现的。

3.3.1 M 文件 S 函数的实现

对于 M 文件 S 函数,SIMULINK 通过 flag 参数告诉 S 函数当前的仿真阶段,以便调用相应的子函数。

为了方便用户编写 M 文件 S 函数,SIMULINK 在 matlabroot \ toolbox \ simulink \ blocks 中提供了一个名称为 sfuntmpl.m 的模板文件。通过该模板文件编写 S 函数,用户只需把自己编写的代码保存到与每个 flag 参数对应的 S 函数方法中即可。

表 3-2 列出了与各个仿真阶段对应的 S 函数的方法,以及与 M 文件 S 函数对应的 flag 参数的值。这样,在进行仿真时,通过传递 flag 参数的值,SIMULINK、M 文件 S 函数和求解解法器将实现相互间的交互,以完成各个仿真阶段特定的操作。

表 3-2　各个仿真阶段对应的 S 函数方法及 flag 值

仿真阶段	S 函数方法	FLAG(M 文件 S 函数)
初始化阶段	MDLINITIALZESIZE	0
下一个采样点计算阶段	MDLGETTIMEOFNEXTVARHIT	4
输出值计算阶段	MDLOUTPUTS	3
更新离散状态阶段	MDLUPDATE	2
微分计算阶段	MDLDERIVATIVES	1
结束任务阶段	MDLTERMONATE	9

通过 sfuntmpl.m 模板文件编写 M 文件 S 函数不仅条理清楚,而且编写简单、方便。因此,建议用户在编写 M 文件 S 函数时使用 sfuntmpl.m 模板文件来编写。模板文件 sfuntmpl.m 的程序代码如下:

```
function [sys,x0,str,ts] = sfuntmpl(t,x,u,flag)
% The following outlines the general structure of an S-function.
switch flag,
  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
  %%%%%%%%%%%%%%%
  % Derivatives %
  %%%%%%%%%%%%%%%
  case 1,
    sys=mdlDerivatives(t,x,u);
  %%%%%%%%%%
  % Update %
  %%%%%%%%%%
  case 2,
    sys=mdlUpdate(t,x,u);
```

```
%%%%%%%%%%%
% Outputs %
%%%%%%%%%%%
case 3,
  sys=mdlOutputs(t,x,u);
%%%%%%%%%%%%%%%%%%%%%%%
% GetTimeOfNextVarHit %
%%%%%%%%%%%%%%%%%%%%%%%
case 4,
  sys=mdlGetTimeOfNextVarHit(t,x,u);
%%%%%%%%%%%%%
% Terminate %
%%%%%%%%%%%%%
case 9,
  sys=mdlTerminate(t,x,u);
%%%%%%%%%%%%%%%%%%%%
% Unexpected flags %
%%%%%%%%%%%%%%%%%%%%
otherwise
  error(['Unhandled flag = ',num2str(flag)]);
end
% end sfuntmpl
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=============================================================================
function [sys,x0,str,ts]=mdlInitializeSizes
% call simsizes for a sizes structure, fill it in and convert it to a
% sizes array.
% Note that in this example, the values are hard coded. This is not a
% recommended practice as the characteristics of the block are typically
% defined by the S-function parameters.
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 0;
sizes.NumInputs      = 0;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;  % at least one sample time is needed
sys = simsizes(sizes);
% initialize the initial conditions
x0  = [];
% str is always an empty matrix
```

```
str = [];
% initialize the array of sample times
ts  = [0 0];
% end mdlInitializeSizes
%===========================================================================
  % mdlDerivatives
% Return the derivatives for the continuous states.
%===========================================================================
function sys=mdlDerivatives(t,x,u)
sys = [];
% end mdlDerivatives
%===========================================================================
% mdlUpdate
% Handle discrete state updates, sample time hits, and major time step
% requirements.
%===========================================================================
function sys=mdlUpdate(t,x,u)
sys = [];
% end mdlUpdate
%===========================================================================
% mdlOutputs
% Return the block outputs.
%===========================================================================
function sys=mdlOutputs(t,x,u)
sys = [];
% end mdlOutputs
%===========================================================================
% mdlGetTimeOfNextVarHit
% Return the time of the next hit for this block. Note that the result is
% absolute time. Note that this function is only used when you specify a
% variable discrete-time sample time [-2 0] in the sample time array in
% mdlInitializeSizes.
%===========================================================================
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;    %  Example, set the next hit to be one second later.
sys = t + sampleTime;
% end mdlGetTimeOfNextVarHit
%===========================================================================
% mdlTerminate
% Perform any end of simulation tasks.
%===========================================================================
function sys=mdlTerminate(t,x,u)
sys = [];
```

```
% end mdlTerminate
```

在这里需要说明的是,sfuntmpl.m 模板文件只是 SIMULINK 为了方便用户编写 M 文件 S 函数而提供的一种 S 函数编写的参考格式,而不是语法要求,用户完全可以对 M 文件 S 函数的子函数名称进行修改或是把子函数代码写入主函数中。此外,当 S 函数需要输入附加参数时,只需向程序的输入参数列表中添加这些参数即可。一般情况下,如果使用 sfuntmpl.m 模板文件进行 S 函数的编写,主函数不需要做任何修改。

从上面的程序代码可以看出,子函数 mdlInitializeSizes 的语法最为复杂,且最为重要。它提供了 S 函数的说明信息,主要包括连续和离散状态的数目、输入和输出的数目以及采样时间的数目等初始条件,它是由 sizes 数组给出的。用户可以通过以下语句得到 sizes 数组的结构。

```
>> sizes = simsizes
sizes =
    NumContStates: 0
    NumDiscStates: 0
        NumOutputs: 0
         NumInputs: 0
    DirFeedthrough: 0
    NumSampleTimes: 0
```

从上面的结果可以看出,sizes 是具有 6 个字段的结构数组,用户可以根据自己的需要输入各个字段的值。各个字段的意义如表 3-3 所示。

表 3-3 sizes 数组各字段意义说明

SIZES 字段名称	意义说明
NUMCONTSTATES	连续状态的数目
NUMDISCSTATES	离散状态的数目
NUMOUTPUTS	输出的数目
NUMINPUTS	输入的数目
DIRFEEDTHROUGH	有无直接馈入(0 表示没有直接馈入,1 表示有直接馈入)
NUMSAMPLETIMES	采样时间的数目

从表 3-3 可以看出,各个字段的意义十分明确,因此在编写 S 函数时要为这些字段赋值非常容易。例如,实例 timestwo.m 中,它的 mdlInitializeSizes 子程序的代码如下:

```
%==========================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%==========================================================
function [sys,x0,str,ts] = mdlInitializeSizes()
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = -1;   % dynamically sized
sizes.NumInputs      = -1;   % dynamically sized
```

```
sizes.DirFeedthrough = 1;  % has direct feedthrough
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
str = [];
x0  = [];
ts  = [-1 0];  % inherited sample time
% end mdlInitializeSizes
```

S函数timestwo的功能是将输入信号乘以2后输出，这里只是对输入信号进行操作，因此，把连续和离散状态的数目都设置为0；而将输入、输出设置为-1，表示它们受驱动模块的控制，是动态可变的；将DirFeedthrough字段的值设置为1，表示有直接馈入；将NumSampleTimes设置为1，表示变量ts的行数为1。

在S函数中，四个输入参数t、x、u和flag都是必须的，并且次序不能更改。同样，SIMULINK也需要四个返回参数：sys、x0、str和ts，这四个返回参数的次序也要按模板指定的顺序排列。输入参数和返回参数的具体意义如下：

- t：表示当前的仿真时间，它用于决定下一个采样时刻，在多采样率时它表示不同的采样时刻点；
- x：表示状态变量，它在任何时候都不能省略；
- u：表示输入向量；
- flag：当flag取不同的值时，表示处在不同的仿真阶段，同时调用不同的S函数的子函数；
- sys：是一个通用的返回参数，它的意义取决于flag的取值。例如，当flag=0时，sys参数返回的是S函数的说明信息；
- x0：是状态向量的初始值，这个返回参数只有当flag=0时才有效；
- str：没有实际意义，它只是为了将来的应用所作的保留参数，在M文件S函数中通常将它设置为空矩阵[ ]；
- ts：是一个m*2的矩阵，用于表示采样时间间隔和采样时间偏移。

3.3.2　M文件S函数实例

下面以timestwo为例，介绍如何通过S函数创建把不同的输入信号扩大2倍后输出的仿真系统。

例3-4　*在输入信号为正弦信号和脉冲信号的作用下，通过timestwo这个M文件S函数把它们分别扩大2倍后输出。*

例3-4的系统仿真模型如图3-5所示，S函数的程序代码在前面已经给出，读者可以参看相应的内容。仿真结果如图3-6所示。实际上timestwo这个S函数的功能相当于SIMULINK模块库中的增益模块。

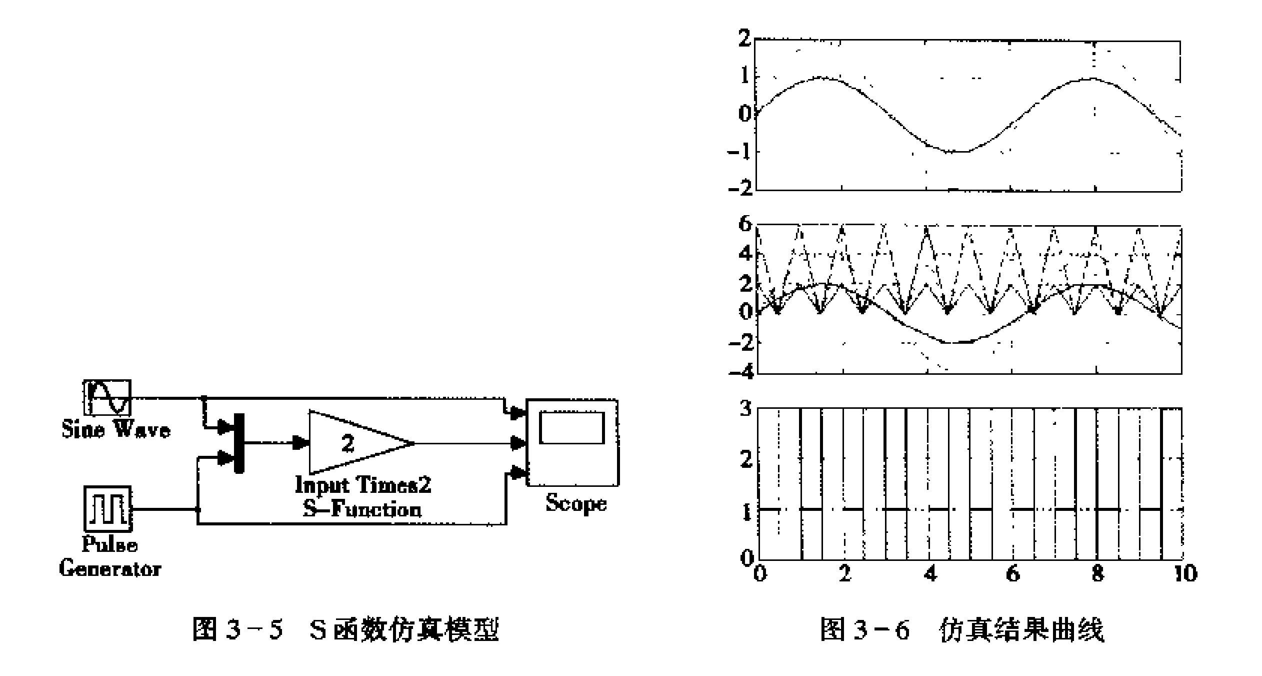


图3-5　S函数仿真模型　　　　图3-6　仿真结果曲线

3.4　C/C++语言S函数

M文件S函数在进行仿真时,不需要进行编译就能够执行。而用C/C++语言编写的S函数,执行时需要首先进行编译,以生成一个动态链接库(文件的扩展名为DLL)文件。C/C++语言S函数有着与M文件S函数相同的结构,但与M文件S函数不同的是,C/C++语言S函数的方法是由C/C++语言函数实现的,而且M文件S函数所能调用的S函数的子函数都可以在C/C++语言S函数中找到等价的子函数,但是反过来则不能,因为C/C++语言S函数提供了更多的方法,功能更加强大。例如,为了不影响仿真速度,可以生成实时代码,可以利用RTW(实时工作空间Real Time Workshop)提供的许多实时功能等。下面对C/C++语言S函数进行介绍。

3.4.1　C语言S函数

C语言S函数,同M文件S函数一样,在进行仿真时需要提供给SIMULINK必要的说明信息,然后通过SIMULINK、解法器和MEX(编译文件)文件进行交互以实现特定的操作,这些操作主要包括初始化、离散状态、计算微分和输出等,它们的定义与M文件S函数中的定义是一样的,如表3-2所示。

为了方便用户编写C语言S函数,与M文件S函数一样,SIMULINK提供了sfuntmpl.c模板文件以及比较复杂的sfuntmpl_doc.c模板文件,这两个模板文件均可以在matlabroot\toolbox\simulink\src目录中找到,C语言S函数的模板文件sfuntmpl.c程序代码如下:

```
* sfuntmpl_basic.c: Basic 'C' template for a level 2 S-function.
#define S_FUNCTION_NAME   sfuntmpl_basic
```

```
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"
/*=============================*
 * S-function methods *
 *=============================*/
/* Function: mdlInitializeSizes ===============================================
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);  /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return;
    }
    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);
    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 1);
    ssSetInputPortRequiredContiguous(S, 0, true); /* direct input signal access */
    ssSetInputPortDirectFeedThrough(S, 0, 1);
    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 1);
    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, 0);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);
    ssSetOptions(S, 0);
}
/* Function: mdlInitializeSampleTimes =========================================
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}
#define MDL_INITIALIZE_CONDITIONS
/* Change to #undef to remove function */
#if defined(MDL_INITIALIZE_CONDITIONS)
/* Function: mdlInitializeConditions ==========================================
    static void mdlInitializeConditions(SimStruct *S)
    {
    }
#endif /* MDL_INITIALIZE_CONDITIONS */
#define MDL_START  /* Change to #undef to remove function */
```

```
#if defined(MDL_START)
  /* Function:mdlStart ==================================================
    static void mdlStart(SimStruct *S)
  {
  }
#endif /*   MDL_START */
/* Function:mdlOutputs ==================================================
static void mdlOutputs(SimStruct *S, int_T tid)
{
     const real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
     real_T       *y = ssGetOutputPortSignal(S,0);
     y[0] = u[0];
}
#define MDL_UPDATE  /* Change to #undef to remove function */
#if defined(MDL_UPDATE)
  /* Function:mdlUpdate ==================================================
    static void mdlUpdate(SimStruct *S, int_T tid)
  {
  }
#endif /* MDL_UPDATE */
#define MDL_DERIVATIVES  /* Change to #undef to remove function */
#if defined(MDL_DERIVATIVES)
  /* Function:mdlDerivatives ==============================================
    static void mdlDerivatives(SimStruct *S)
  {
  }
#endif /* MDL_DERIVATIVES */
/* Function:mdlTerminate ==================================================
static void mdlTerminate(SimStruct *S)
{
}
/*=====================================================================*
 * See sfuntmpl_doc.c for the optional S-function methods *
 *=====================================================================*/
/*================================*
 * Required S-function trailer *
 *================================*/
#ifdef  MATLAB_MEX_FILE     /* Is this file being compiled as a MEX-file? */
#include "simulink.c"       /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"        /* Code generation registration function */
#endif
```

从上面的模板文件的程序代码可以看出，与 M 文件 S 函数不同，在 C 语言 S 函数中，

SIMULINK不是通过flag参数来调用每个仿真阶段的子函数(即S函数方法),而是在适当的时候直接调用相应的子函数。

下面以timestwo.c为例对C语言S函数进行介绍。在介绍之前,首先把timestwo.c文件更名为twotimes.c,这主要是为了区分timestwo.m文件,虽然在调用优先级上,C语言S函数timestwo.c优先于M文件S函数timestwo.m,但其调用格式是一样的,为了避免混淆,对文件进行改名是必要的。另外,当timestwo.c生成一个timestwo.dll编译文件之后,读者就不能再使用命令mex对其进行编译,为了让读者了解mex对其命令的使用方法,对文件改名称也是必要的。twotimes.c程序代码如下:

```
#define S_FUNCTION_NAME  twotimes
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"
/*=================*
* Build checking *
*=================*/
/* Function:mdlInitializeSizes=============================================
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
    }
    if (! ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, DYNAMICALLY_SIZED);
    ssSetInputPortDirectFeedThrough(S, 0, 1);
    if (! ssSetNumOutputPorts(S,1)) return;
    ssSetOutputPortWidth(S, 0, DYNAMICALLY_SIZED);
    ssSetNumSampleTimes(S, 1);
    /* Take care when specifying exception free code - see sfuntmpl_doc.c */
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE |
                 SS_OPTION_USE_TLC_WITH_ACCELERATOR);
}
/* Function:mdlInitializeSampleTimes=======================================
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}
/* Function:mdlOutputs=====================================================
static void mdlOutputs(SimStruct *S, int_T tid)
{
    int_T             i;
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
```

```
    real_T          *y     = ssGetOutputPortRealSignal(S,0);
    int_T           width = ssGetOutputPortWidth(S,0);
    for (i=0; i<width; i++) {
        *y++ = 2.0 *(*uPtrs[i]);
    }
}
/* Function:mdlTerminate ===========================================
static void mdlTerminate(SimStruct * S)
{
}
#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

下面对 twotimes.c 文件进行如下说明:

- 这个文件定义了 S 函数的名称(twotimes)、格式(level 2),还包含了 simstruct.h 头文件。在 M 文件 S 函数中,通过返回变量参数 sys 来返回各个子函数的结果,而在 C 语言 S 函数中,各个子函数通过对数据结构 simstruct 进行存取来完成与 SIMULINK 的交互。因此,需要包含 simstruct.h 这个头文件。
- **mdlInitializeSizes 子函数**:用于定义在 twotimes.c 中 S 函数的说明信息。这些说明信息的定义都是通过 SIMULINK 接口函数读取 simstruct 数据结构完成的,这些说明信息主要包括:

  零参数:即把 S 函数的附加参数设置为 0。这时,若在 S 函数对话框的参数文本框中输入参数,则会发出出错警告信息。

  一个输入端口和一个输出端口:在这里,不仅定义了一个输入端口和一个输出端口,而且还定义了它们均属于动态可变的。同时,输入端口是直接馈入,因此需要进行如下设置:ssSetInputPortDirectFeedThrough(S, 0, 1),其中,0 表示没有直接馈入,1 表示有直接馈入。

  指定采样时间:与 M 文件 S 函数不同,在 C 语言 S 函数中,在 mdlInitializeSizes 子函数的 ssSetNumSampleTimes 中设置采样时间个数,而其采样时间值需要在 mdlInitializeSampleTimes 子函数中设置。

  代码设置为无异常(exception free):当指定代码无异常时会加快仿真速度。
- **mdlInitializeSampleTimes 子函数**:用于设定采样时间和时间偏移的实际值。在本例中采样时间从驱动模块继承,时间偏移为 0。
- **mdlOutputs 子函数**:用于将输入函数值扩大 2 倍,并把计算结果作为输出信号输出。
- **mdlTerminate 子函数**:该子函数的作用是终止仿真,然后执行用户指定的操作。在本例中仿真结束时没有指定执行任何操作,所以将其设置为空。
- S 函数最后指定这个代码关联的是 SIMULINK 代码,还是 RTW 代码。在本例

中代码关联的是SIMULINK代码。

用户如果需要建立更加复杂的C语言S函数,可以使用sfuntmpl_doc.c模板文件进行S函数的编写,有关sfuntmpl_doc.c模板文件,用户可以参看帮助文件或是S函数的demos文件,这里不再作介绍。

3.4.2　C++语言S函数

C++语言S函数编写的过程和执行方法以及模板文件与C语言S函数的编写过程和执行方法以及模板文件基本上是一样的,只不过它的模板文件的扩展名是.cpp而不是.c。此外,C++语言S函数与C语言S函数的区别在于模板文件把S函数的方法(即子函数)添加到了extern "c"语句中。这是因为SIMULINK仿真引擎假设所调用的子函数遵循的是C语言规则。在编译C++语言S函数时,C++编译器将会从extern "c"语句获取信息,使得编译器产生与SIMULINK兼容的规则方法以实现子函数的调用。在matlabroot\toolbox\simulink\src目录下可以找到sfun_counter_cpp.cpp,它包含了extern "c"语句的使用方法。sfun_counter_cpp.cpp程序代码如下:

```
/*  File     : sfun_counter_cpp.cpp
#include "iostream.h"
class  counter {
    double  x;
public:
    counter() {
        x = 0.0;
    }
    double output(void) {
        x = x + 1.0;
        return x;
    }
};
#ifdef __cplusplus
extern "C" { // use the C fcn-call standard for all functions
#endif        // defined within this scope
#define S_FUNCTION_LEVEL 2
#define S_FUNCTION_NAME  sfun_counter_cpp
#include "simstruc.h"
/*===========================*
 * S-function methods *
 *===========================*/
/* Function:mdlInitializeSizes ================================================
static void mdlInitializeSizes(SimStruct * S)
{
    ssSetNumSFcnParams(S, 1);  /* Number of expected parameters */
```

```
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return;
    }
    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);
    if (!ssSetNumInputPorts(S, 0)) return;
    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 1);
    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, 0);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 1); // reserve element in the pointers vector
    ssSetNumModes(S, 0); // to store a C++ object
    ssSetNumNonsampledZCs(S, 0);
    ssSetOptions(S, 0);
}
/* Function:mdlInitializeSampleTimes ==================================
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, mxGetScalar(ssGetSFcnParam(S, 0)));
    ssSetOffsetTime(S, 0, 0.0);
}
#define MDL_START   /* Change to #undef to remove function */
#if defined(MDL_START)
  /* Function:mdlStart ==================================================
  static void mdlStart(SimStruct *S)
  {
      ssGetPWork(S)[0] = (void *) new counter; // store new C++ object in the
  }                                             // pointers vector
#endif /*   MDL_START */
/* Function:mdlOutputs ================================================
static void mdlOutputs(SimStruct *S, int_T tid)
{
    counter *c = (counter *) ssGetPWork(S)[0];   // retrieve C++ object from
    real_T  *y = ssGetOutputPortRealSignal(S,0); // the pointers vector and use
    y[0] = c->output();                          // member functions of the
}                                                // object
/* Function: mdlTerminate ============================================== static
void mdlTerminate(SimStruct *S)
{
    counter *c = (counter *) ssGetPWork(S)[0]; // retrieve and destroy C++
    delete c;                                  // object in the termination
}                                              // function
```

```
/*========================================================*
 * See sfuntmpl.doc for the optional S-function methods *
 *========================================================*/
/*========================*
 * Required S-function trailer *
 *========================*/
#ifdef  MATLAB_MEX_FILE      /* Is this file being compiled as a MEX-file? */
#include "simulink.c"        /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"         /* Code generation registration function */
#endif
#ifdef __cplusplus
} // end of extern "C" scope
#endif
```

从 sfun_counter_cpp.cpp 程序代码可以看出,定义 S 函数的调用方法都包含在它的 extern "c"语句中,用户可以根据需要把相应的代码填入其中。

3.5　Stateflow 原理与应用

Stateflow 是有限状态机(finite state machine)的图形工具,它可以用于解决复杂的逻辑问题,用户可以通过图形化工具实现在不同状态之间的转换。Stateflow 可以直接嵌入到 SIMULINK 仿真模型中,并且在仿真的初始化阶段,SIMULINK 会把 Stateflow 绘制的逻辑图形通过编译程序转换成 C 语言 S 函数,使二者有机地结合在一起。Stateflow 可以在 SIMULINK Extra 模块库中找到。

3.5.1　Stateflow 原理

Stateflow 的仿真原理是有限状态机(finite state machine)理论,有限状态机是指系统含有可数的状态,在相应的状态事件发生时,系统会从当前状态转移到与之对应的状态。在有限状态机中实现状态的转移是有一定条件的,同样相互转换的状态都会有状态转移事件,这样就构成了状态转移图。在 SIMULINK 的仿真窗口中,允许用户建立有限个状态以及状态转移的条件与事件,从而绘制出有限状态机系统,这样就可以实现对系统的仿真。Stateflow 的仿真框图一般都会嵌入到 SIMULINK 仿真模型中,同时实现状态转移的条件或是事件既可以取自 Stateflow 仿真框图,也可以来自 SIMULINK 仿真模型。

3.5.2　Stateflow 应用基础

1. 创建 Stateflow 仿真框图

在 MATLAB 环境下,要创建一个新的 Stateflow 仿真框图,只需要在 MATLAB 命令

窗口输入命令：

```
>> sfnew
```

执行命令后会弹出如图 3－7 所示的模型框图窗口，其中 Chart 是 Stateflow 空白的模块图标，用鼠标双击“Chart”模块，则会弹出如图 3－8 所示的 Stateflow 的模型框图编辑窗口。Stateflow 提供了强大的图形编辑功能，因此在这里用户可以根据需要编辑各种 Stateflow 的模型框图。



图 3－7　Stateflow 模型窗口

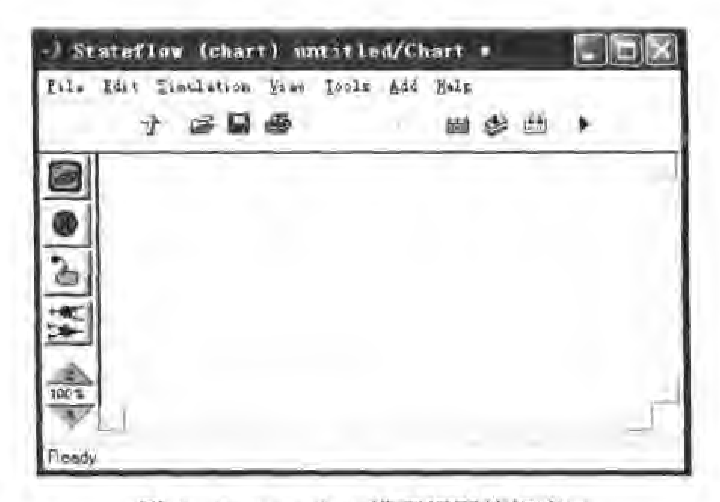


图 3－8　Stateflow 模型框图编辑窗口

2．Stateflow 仿真框图编辑工具

在图 3－8 所示的 Stateflow 仿真模型框图编辑窗口中，左侧提供了 Stateflow 模型框图的各种编辑工具。其中包括：

- 状态编辑工具。
- 历史节点编辑工具。
- 默认的状态转移编辑工具。
- 可连接的节点编辑工具。
- 比例缩放编辑工具。

接下来分别对各个编辑工具进行介绍。

1）状态编辑工具

在 Stateflow 模型编辑窗口中，创建一个状态很简单，只需要选定“状态编辑”工具（），然后将其拖到 Stateflow 模型编辑窗口中即可，这时在 Stateflow 模型框图编辑窗口

中将出现一个状态模块，并且在模块的左上方出现一个问号，如图 3－9 所示。

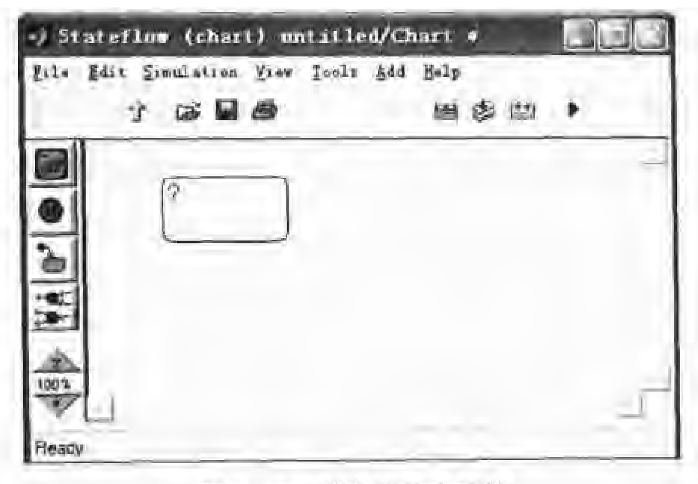


图 3－9　创建的状态模块

一个状态有两种行为：激活与非激活。右击所创建的状态模块，在弹出的快捷菜单中选择 properties 命令，则会弹出状态属性设置对话框，如图 3－10 所示，在这里用户可以设置状态模块的属性。



图 3－10　状态属性设置对话框

在 Label 文本框中可以设置状态模块的标识以及是否激活。定义状态激活有如下方法：

- Entry 激活：表示由于发生状态转移而进入此状态时，有限状态机将激活并进入激活语句或事件。
- On Event Action 激活：表示在发生某个事件时激活某个语句或事件。
- During Action 激活：表示在进入该状态而尚未退出时，则在此状态或期间激活某个语句或事件。
- Exit Action 激活：表示由于从该状态退出而激活某个语句或事件。
- 输出到 SIMULINK 激活：通过状态流模块和数据口，当输出到 SIMULINK 时激活此状态。

2）节点编辑工具

节点编辑工具包括可连接节点编辑工具和历史节点编辑工具。其中，利用可连接节

点编辑工具可以实现同一连接点基于多个条件或事件,可将一个状态转移到多个状态;历史节点编辑工具用于标识与历史的联系。当进入到此级时,最后被激活的状态将变成下一次进入时被激活的状态。

3)状态转移编辑工具

通过状态转移编辑工具从一个状态的边界画线到另一个状态的边界,即可实现状态转移。

在状态转移线上可以编辑引起状态转移的事件,方法如下:右击状态转移线,在弹出的快捷菜单中选择 Properties 命令,在弹出的状态对话框的 Label 文本框中输入状态转移表达式。表达式的格式如下:

envent[condition]{condition action}/transition action

下面对表达式中各个参数分别进行说明。

- envet:当程序执行时判断引起状态转移的 event(事件)是否为真,为真时,则具备了状态转移的条件。
- [condition]:当此条件成立时,激活{condition action}而且发生状态转移。
- /transition action:从源状态退出后进入目标状态前执行这个语句。

状态转移线共有以下三种:

- 从一个状态到另一个状态(或节点)的转移线;
- 自环转移线;
- 默认的状态转移线(没有源状态,只有目标状态)。

此外,在 Stateflow 模型框图中实现状态转移时,可以使用时间逻辑运算符,如 after、before、at 和 every,这些运算符的使用规则十分简单,值得读者注意的是,时间逻辑运算符只能出现在状态转移的条件和状态激活上。

4)比例缩放编辑工具

使用比例缩放编辑工具,可以使已经建立的 Stateflow 模型框图放大和缩小,这样可以增强其可读性。

3. Stateflow 仿真框图数据和事件的设置

在创建 Stateflow 仿真模型框图时,还需要对数据、事件等进行定义,实现它们的定义的方法有两种,一种是使用 Stateflow 编辑器,另一种是使用 Stateflow 浏览器。这里只介绍采用第一种方法,具体的定义方法如下:首先,在 Stateflow 编辑器窗口的工具栏菜单中选择 Add/Data(数据)(或 Event(事件))命令,这就选择了定义的是 Data,还是 Event;然后再指定它们的类型,它们的类型如表 3-4 所示;最后,利用数据或是事件参数设置对话框指定它们的选项。

表 3-4　数据和事件的类型

数据(DATA)	事件(EVENT)
LOCAL	LOCAL
INPUT FROM SIMULINK	INPUT FROM SIMULINK
OUTPUT TO SIMULINK	OUTPUT TO SIMULINK
TEMPORARY	
CONSTANT	

4．Stateflow 仿真框图输入输出设置

对于 Stateflow 仿真模型框图一般还需要进行输入输出的设置，以达到与 SIMULINK 进行数据交互和设置 Stateflow 模型框图输入输出端子的目的。当选择数据类型是 Input from SIMULINK 时，则会将此数据设置成 Stateflow 模型框图的输入端子，输入信号设置对话框如图 3－11 所示。类似的，当选择数据类型为 Output to SIMULINK 时，则会将数据设置成 Starteflow 模型框图的输出端子，输出信号设置对话框与输入信号设置对话框基本相同，只是 Scope(数据类型)为 Output to SIMULINK。



图 3－11　输入信号设置对话框

3.5.3　Stateflow 应用实例

随着系统的复杂化，在进行系统仿真时使用 Stateflow 的情况越来越多。Stateflow 能够实现有限状态机系统的仿真，图形化的建模环境使仿真的实现十分方便。下面通过一个的实例介绍 Stateflow 的建模与应用。

例 3－5　利用 SIMULINK 和 Stateflow 模型框图实现汽车自动变速控制系统的建模和仿真。

汽车自动变速系统模拟汽车从起步到稳定运行的状态过程。本例通过添加两个速度限来实现汽车系统的加、减档，即系统的自动变速。对于汽车自动变速系统的数学模型在这里就不再介绍，这里只介绍如何实现它的建模以及速度变换逻辑。汽车自动变速系统的 SIMULINK 仿真模型如图 3－12 所示。

从如图 3－12 所示的系统仿真模型中可以看出，在创建系统仿真模型时，把汽车本身以及它的发动机、传动装置、速度阈值计算模块都建成了子系统模块。汽车本身的子系统模块的仿真模型如图 3－13 所示，发动机、传动装置以及阈值计算子系统模块的仿真模型分别如图 3－14、3－15 和 3－16 所示。

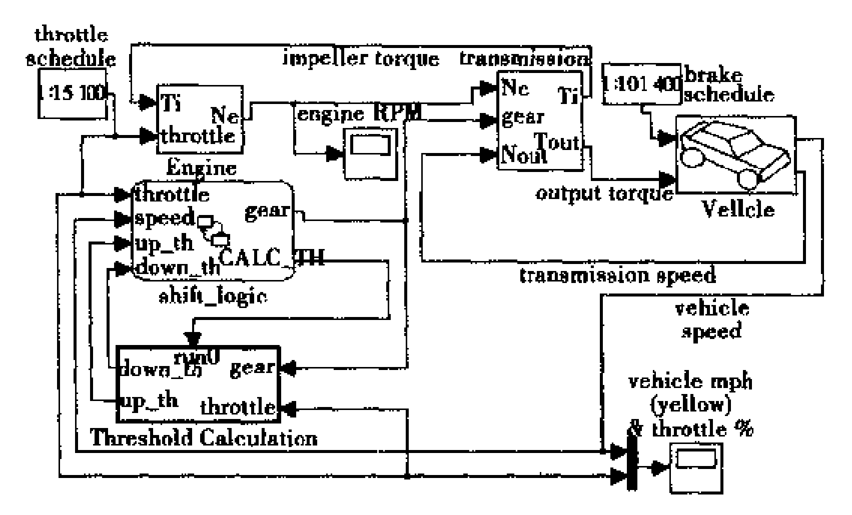


图 3-12 汽车自动变速系统仿真模型

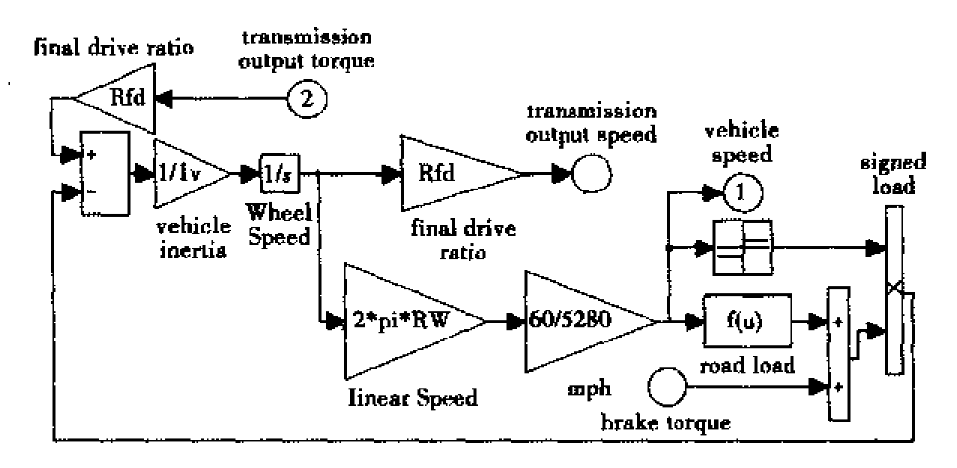


图 3-13 汽车子系统模块仿真模型

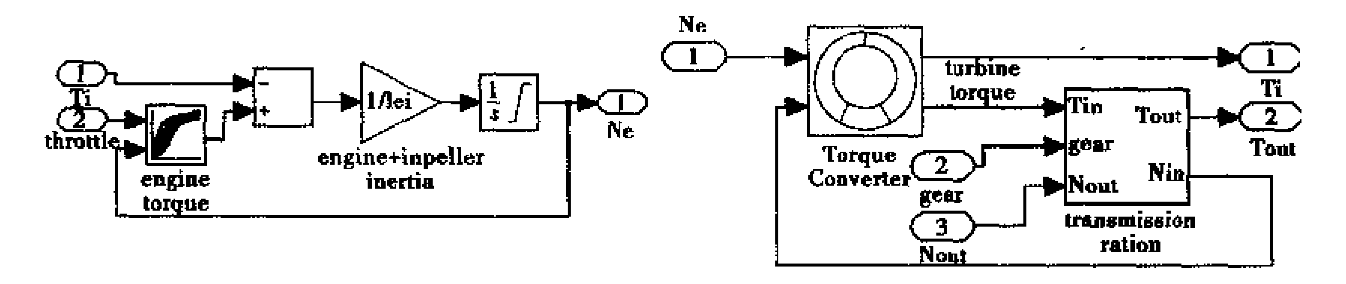


图 3-14 发动机子系统模块仿真模型　　图 3-15 传动装置子系统模块仿真模型

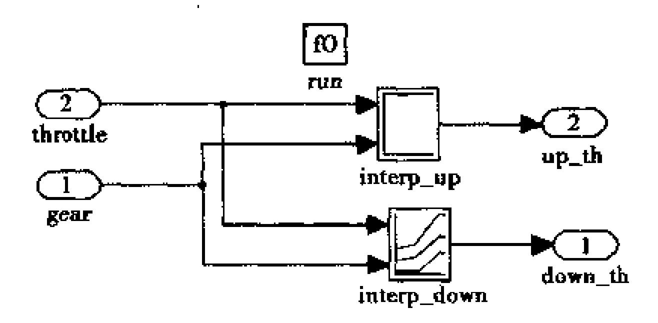


图 3-16 阈值计算子系统模块仿真模型

由图 3－15 可以看出，传动装置子系统模块中还包括 Torque Converter（力矩转换器）子系统模块和 transmission ratio（传动比率）子系统模块，它们的仿真模型如图 3－17 所示。

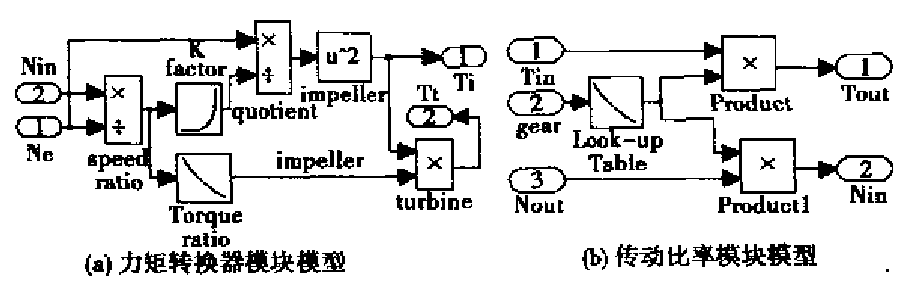


图 3－17　力矩转换器和传动比率模块仿真模型

通过图 3－18 可以将该例中的汽车状态转化逻辑用 Stateflow 模型框图表示出来。然后再根据 Stateflow 模型框图需要，利用前面介绍的方法定义它的数据和事件类型，如图 3－19 所示。

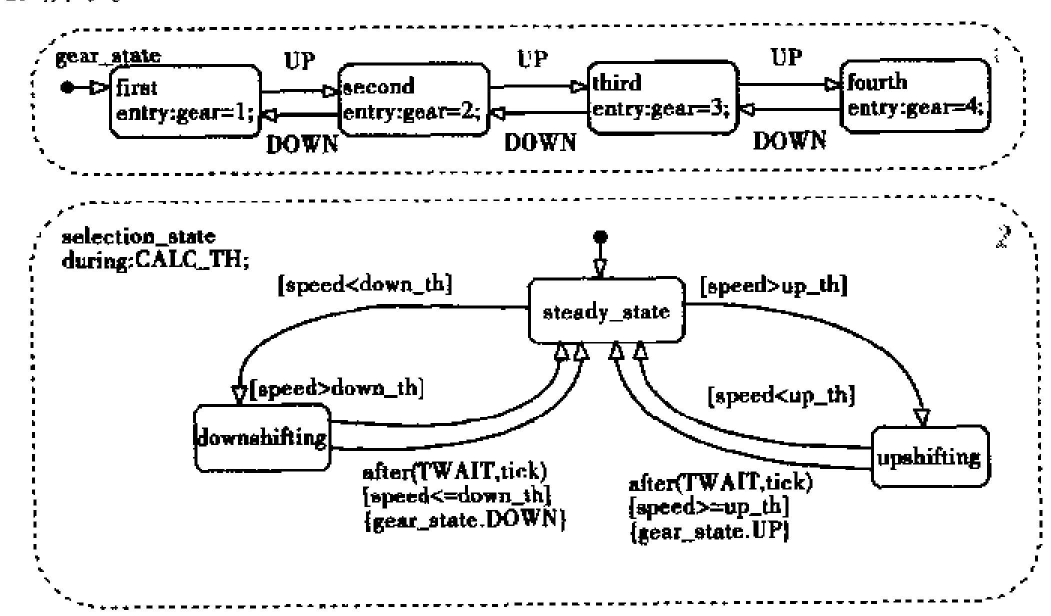


图 3－18　汽车速度转换 Stateflow 模型框图

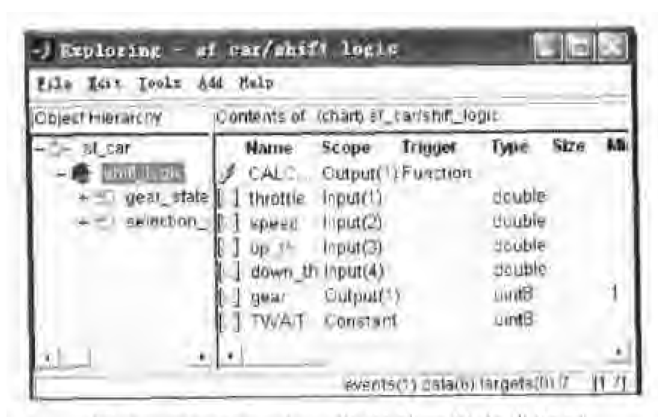


图 3－19　Stateflow 数据和事件定义窗口

汽车自动变速系统的仿真模型建立完毕后，利用第 2 章中介绍的仿真运行方法运行


仿真,得到的结果如图 3－20 所示。

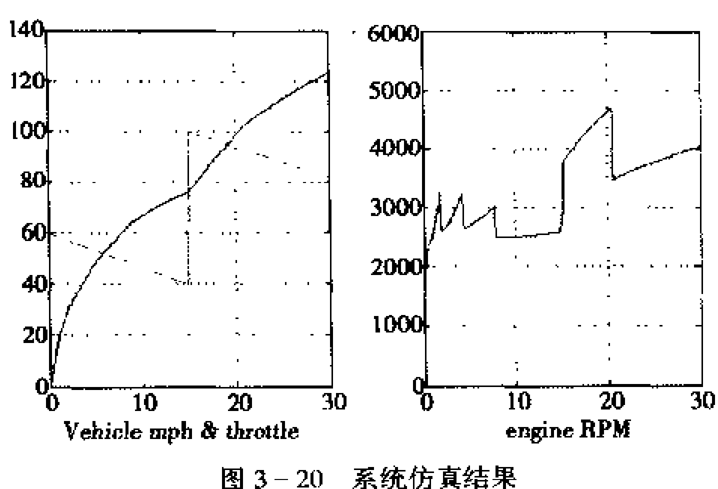


图 3－20　系统仿真结果

3.6　基于 SIMULINK 的 VR 技术

在本节中将介绍在 MATLAB 6.1 中新增加的一个工具箱——虚拟现实(Virtual Reality,VR)技术工具箱。希望通过对这一节的学习,读者能够对 VR 技术及其应用有一个初步的了解,为将来更深层次地学习和应用 VR 技术奠定坚实的基础。

3.6.1　VR 技术介绍

VR 技术就是把计算机从善于处理数字化的单维信息改变为处理人所能感觉到的、在思维过程中所能接触到的、除了数字化信息之外的其他各种表现形式的多维信息。VR 技术近些年来在技术研究领域十分活跃,它汇集了计算机图形学、多媒体技术、人工智能、人机接口技术、传感器技术、高度并行的实时计算技术和人的行为学研究等多项关键技术。目前,VR 技术已在军事、医学和商业等领域广泛应用。

MATLAB 6.1 在 MATLAB 以前版本的基础上增加了 VR 技术工具箱,用于在 MATLAB 仿真环境下实现 VR 技术的仿真与应用。VR 技术工具箱将 MATLAB 与 SIMULINK 扩展到了虚拟现实图形世界中,它可以在三维虚拟现实环境中提供可视化和可与动态系统交互的解决方案,这些动态系统由 MATLAB 和 SIMULINK 进行描述。VR 技术工具箱扩展了 MATLAB 和 SIMULINK 处理虚拟现实图像的能力。使用标准的 VRML(Virtual Reality Modeling Language)文件可以创建活动的三维场景,并且由 MATLAB 和 SIMULINK 进行驱动。通过这一接口用户可以在虚拟的三维模型中观察动态系统的模拟。VR 技术工具箱中的大多数功能可以由 SIMULINK 模块描述,因此可以选择将 SIMULINK 信号连接到虚拟世界中。VR 技术由可用的 VRML 节点自动搜索虚拟世界来实现。接下来对 VRML 作简要地介绍。

VRML 是一种常用的虚拟现实建模语言,也就是 HTML 的三维模拟,它是由 VRML 浏览器来描述现实世界并且实现与现实世界的链接。VRML 既可以建立虚拟的三维世

界模拟模型，也可以建立真实场景的模型。VRML 文件是虚拟空间的文本描述形式，它是一个由文本编辑器生成的文本文件，文件的扩展名为“.wrl”。VRML 在描述三维空间时，其提供的坐标框架符合右手规则，便于理解。

V-realm builder 是一个编辑 VRML 程序的实用可视化工具，同时，它还是强大的三维物体构造工具包，它生成的三维物体和虚拟世界可以通过 VRML 浏览器进行观察，能够使复杂的物体建模变得十分简单，无论是这个领域的专家还是初学者都能够很容易地学会它的使用方法及编程技巧。在 MATLAB 环境下，就是利用 V-Realm Builder 来实现 VR 技术的应用的。如果用户在安装 MATLAB 6.1 版本时安装了 VR 技术工具箱，在 MATLAB6p1 \ toolbox \ vr \ vrealm \ program 目录下双击 vrbuild2.exe 文件，即可以打开 V-Realm Builder 窗口，如图 3－21 所示。这个窗口中包含了大量的命令按钮和图标，这里不作介绍，用到时可查阅相关的帮助文档。

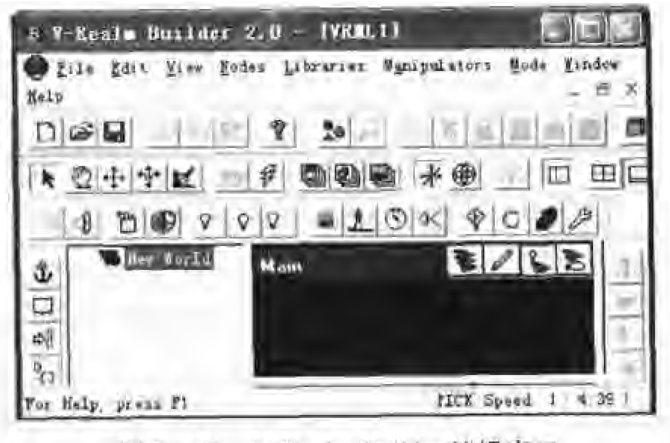


图 3－21　V-Realm Builder 编辑窗口

3.6.2　SIMULINK 下的 VR 技术应用

SIMULINK 4.1 版本提供了 VR 技术工具箱子模块库，如图 3－22 所示。在此工具箱子模块库中，包含了实现 VR 技术的各种模块。

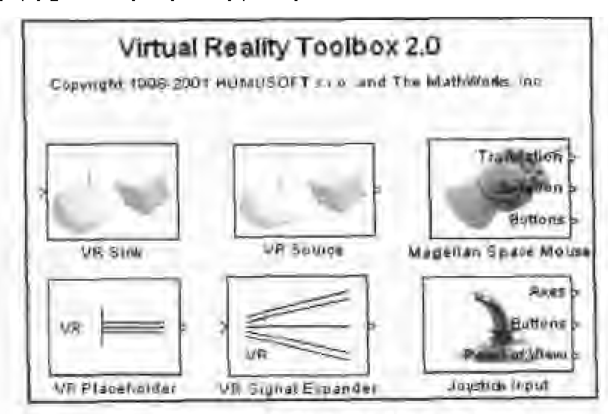


图 3－22　VR 技术子模块库

VR 技术子模块的使用方法与前面介绍的 SIMULINK 模块库中模块的使用方法是一样的。下面通过实例说明 VR 技术子模块的使用方法。

例3-6 这是一个利用VR技术子模块库中的模块实现一个球体在只以外力为重力的作用下运动轨迹的仿真。

实现这个系统的仿真模型如图3-23所示。

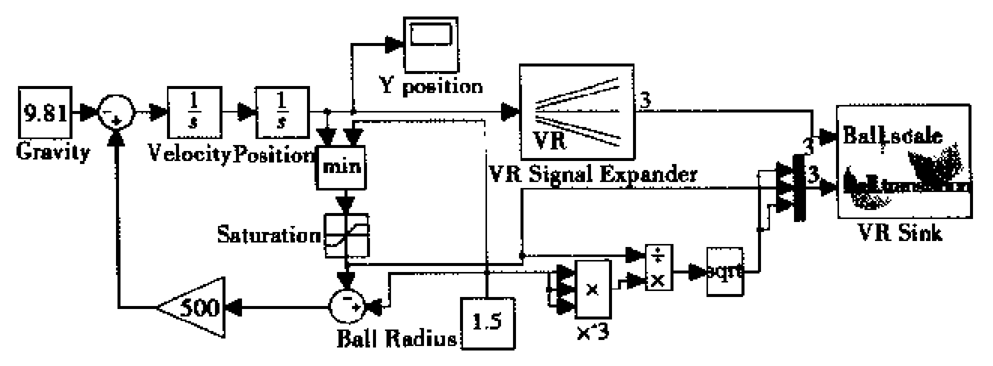


图3-23 系统仿真模型

在这个系统中,使用了VR技术模块库中的两个子模块,即VR Signal Expander(虚拟现实信号扩展模块)和VR Sink(虚拟现实输出模块)。虚拟现实信号扩展模块用于将输入信号扩展到完全符合VRML空间的信号,并使用VR Placeholder Signal (VR占位符信号)在输出端口填满空白位置。虚拟现实输出模块能够接收SIMULINK的信号,并把结果以虚拟现实的形式显示,也可以使用MATLAB的函数view观察描述系统的静态图形。图3-24所示为弹跳的球体的运动位移轨迹。

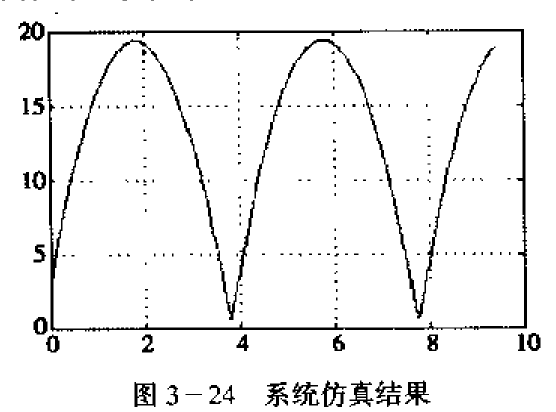


图3-24 系统仿真结果

3.7 本章小结

本章主要介绍了SIMULINK在仿真中的高级应用。绘制和修改SIMULINK仿真模型不仅可以使用SIMULINK模块库中的基本模块实现,而且还可以通过MATLAB语句与命令来实现。

另外,为了更好地发挥SIMULINK的功能,让读者更深一步地了解SIMULINK仿真环境,在本章中还介绍了它的几个扩展工具,主要包括S函数、Stateflow以及VR技术工具箱。掌握了S函数的编写方法和技巧,可以实现更加复杂和具有特定功能的系统仿真,在编写各种形式的S函数时建议读者使用相应的模板文件,它不仅会使S函数的编写变得十分简单,而且它的思路清晰,可以很容易实现模块的功能。Stateflow可以实现复

杂系统的逻辑监控，已得到越来越广泛的应用。VR 技术目前虽然还不是很成熟，但是它已十分受重视，并被广泛地应用于各研究领域，特别是军事领域。MATLAB 6.1 版本通过提供 VR 技术工具箱来实现 MATLAB 仿真环境下 VR 技术的应用。

习　题

1. 使用 MATLAB 语句和命令绘制如图 2-61 所示的 SIMULINK 仿真模型，并查看其模型文件，了解仿真模型及其模块的各个属性。
2. 利用 SIMULINK 模块库中的模块和 MATLAB 语句与命令实现如图 3-25 所示的仿真模型的绘制。

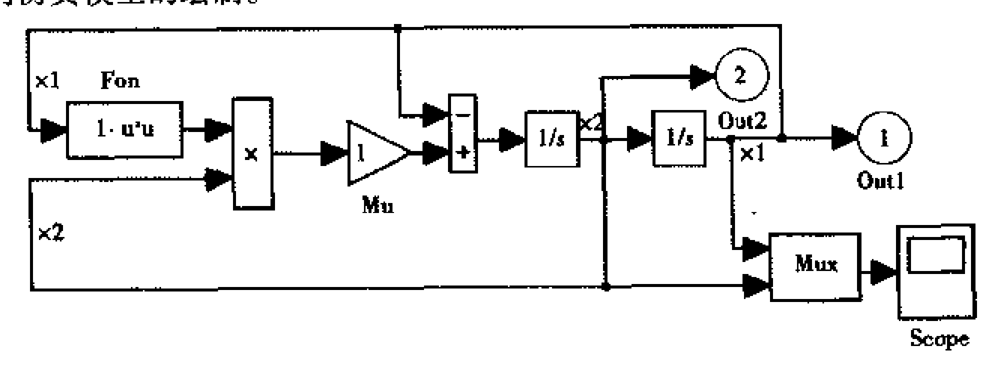


图 3-25　vdp 方程仿真模型

3. 使用 M 文件 S 函数模板文件编写实现积分器模块功能的 S 函数。
4. 使用 C 语言 S 函数模板文件编写实现积分器模块功能的 S 函数。
5. 某仓库日供货量均值为 120，是方差为 25 的正态分布函数，初始库存量为 500，当库存量下降到某一数量时即订货，提出订货后到货延迟天数为均匀分布的随机数，20%货物延迟为两天，50%延迟为三天，30%延迟为四天，库存每件/天的费用是 C1 元，缺货每件/天的费用是 C2 元，一次订货费用为 C3 元，试制定一个订货起点数、每次订货数，以使总的费用最小。并使用实现 Stateflow 模型框图和 SIMULINK 实现对系统的仿真。

第4章　MATLAB 在信号处理仿真中的应用

知识点：

- 仿真模型的创建
- 仿真信号
- 信号的变换域分析
- 数字滤波器的设计
- 信号处理的交互式工具—SPTool
- 信号处理仿真实例
- 数字信号处理仿真模块

本章主要介绍各种常用的离散时间信号、信号的Z颜色变换和傅立叶变换分析、IIR数字滤波器和FIR数字滤波器的结构和设计，以及MATLAB信号处理工具箱中的交互式信号处理工具——SPTool。通过本章的学习，可以了解和掌握常用仿真信号的产生、信号的变换域分析、各种数字滤波器的设计，从而可以完成各种数字信号的分析与处理。

DSP(Digital Signal Processing，数字信号处理)是研究用系统对含有信息的信号进行数字化处理(变换)，以获得人们所希望的信号，从而达到提取信息、便于利用的一门学科。数字信号处理包括对信号进行滤波、变换、检测、谱分析、估计、压缩、识别等一系列的加工处理。随着信息时代、数字世界的到来，数字信号处理已成为一门极其重要的学科和研究领域。

值得高兴的是，随着MATLAB的出现和不断完善，尤其是MATLAB信号分析工具箱的推出，越来越多的工程师和科研人员已经意识到，利用MATLAB解决电子通信领域的实际问题是既省时又省力的事情。由于MATLAB具有计算快而准确、使用方便的优点，在过去的十年中，MATLAB已经成为DSP应用中分析和设计的主要仿真工具。

在本章中，首先讨论信号处理中需要的各种仿真信号，接着讨论信号的变换域分析以及数字滤波器的设计；然后对MATLAB信号处理工具箱中的交互式信号处理工具——SPTool作简要介绍；最后通过具体应用实例来详细说明如何运用MATLAB及其提供的信号处理工具箱进行数字信号处理。

4.1　仿真信号

信号通常分为模拟信号和数字信号两大类。在计算机中，或者说是在MATLAB仿真中，信号都是以离散形式出现的。由于计算机本身以离散方式处理所有数据，因此，只可能产生离散信号；如果要产生连续信号，则只能是让信号的离散时间间隔趋于无穷小。所以这里只针对离散时间信号进行讨论。

模拟信号为连续信号，用 X(t)表示；数字信号为离散信号，用 X(n)表示，其中 n 是整数，表示时间的离散时刻。在 MATLAB 中，数字信号用矩阵表示，一个列向量表示一个有限长序列，即一维信号；一个 n * m 矩阵表示 m 个通道信号，即多维信号。本书主要讨论一维信号。

通常，用一个列向量表示一个信号序列时，还需要用一个对应的列向量表示信号的各个采样时刻。如：

n=[-5:5];　　　　X=[5 3 2 4 0 1 2 3 5 6 7];

当不需要采样位置信息时，可以只使用列向量 X 表示序列。需注意的是，MATLAB 无法表示任意无限长序列。

4.1.1 基本信号序列

下面将工程应用和理论研究中经常用到的数字信号序列(基本信号序列)的数学定义式和 MATLAB 实现语句做如表 4-1 所示的归纳。

表 4-1　基本信号序列

序列	数学表达式	MATLAB 函数表达式
单位冲击序列	$\delta(n)=\begin{cases}1 & n=0\\0 & n\neq 0\end{cases}$	X=[1 ZEROS(1,N-1)];
单位阶跃序列	$u(n)=\begin{cases}1 & n\geqslant 0\\0 & n<0\end{cases}$	X=ONES(1,N);
矩形序列	$R_N(n)=\begin{cases}1 & 0\leqslant n\leqslant N-1\\0 & else\end{cases}$	X=ONES(1,N);
实指数序列	$x(n)=a^n,\forall n,a\in R$	N=0:N-1;X=A.^N;
复指数序列	$x(n)=e^{(\delta+j\omega)n},\forall n\in R$	N=0:N-1;　X=EXP((A+J * W) * N)
随机序列		X=RAND(1,N) 或 X=RANDN(1,N)

注：rand(1,N)产生[0 1]上均匀分布的随机信号。randn(1,N)产生均值为 0、方差为 1 的 Guassian 随机信号序列。

基本信号序列中的单位阶跃序列、矩形序列和随机序列可以在 MATLAB 的 Simulink 环境下产生。产生这三个序列的仿真模块位于 Simulink 子模块库的信号源子模块集中，如图 4-1 所示。

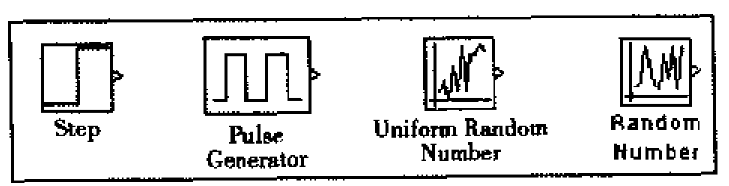


图 4-1　基本信号序列产生模块

提示：在图 4-1 中，Step 模块产生阶跃序列，Pulse Generator 模块产生矩形序列，Uniform

Random Number 模块产生均匀分布的随机序列，Random Number 模块产生指定均值和方差的随机序列。另外，标准 SIMULINK 子模块库中的 Source 子模块集中的 Signal Generator 模块也可以产生随机序列。

4.1.2 其他信号序列

1. 基本周期信号

基本周期信号包括以下几类：

● 方波：MATLAB 工具箱用 square()函数产生方波，其代码如下：

```
t=0:0.1*pi:5*pi;
x=square(t);plot(t,x);
```

提示：Simulink 子模块库的 Source 子模块集中的 Signal Generator 模块中也可以产生方波。

● 正弦波：MATLAB 工具箱用 sin()函数产生正弦波，其代码如下：

```
t=0:0.01*pi:2*pi;
x=sin(t);plot(t,x);
```

提示：Simulink 子模块库的 Source 子模块集中的 Sine Wave 模块和 Signal Generator 模块均可产生正弦波信号。

● 锯齿波：在 MATLAB 工具箱中，sawtooth()函数可以产生锯齿波信号。如图 4-2所示，其代码如下：

```
Fs=500;                          % 抽样频率为500Hz
t=0:1/Fs:0.3;                    % 抽样长度为0.3s
x=sawtooth(2*pi*50*t);           % 信号频率为50Hz
plot(t,x);
axis([0 0.3 -1 1]);              % 画出了0.3秒的波形
```

sawtooth()函数的调用格式为 sawtooth(t, width)，其中，widthd 在 [0,1]之间取值，用于决定峰值顶点出现的偏移量。因此，width 取不同值，此函数可以产生各种三角波信号。

提示：Simulink 子模块库的 Source 子模块集中的 Repeating Sequence 模块也可产生锯齿波。

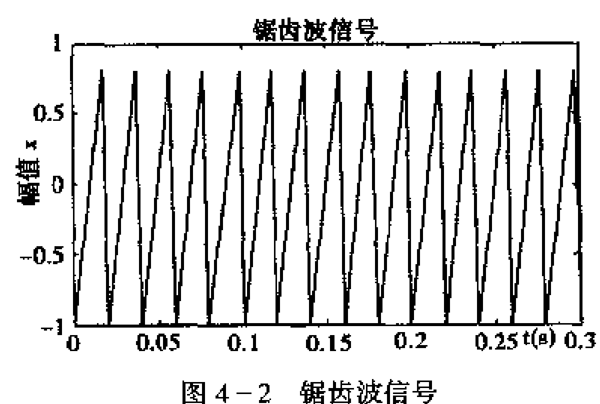


图 4-2　锯齿波信号

2. 采样信号

采样信号的数学定义式为：

$$\mathrm{sinc}(t)=\frac{\sin(\pi t)}{\pi t} \tag{4-1}$$

在MATLAB工具箱中，sinc()函数能够产生采样信号，如图4-3所示。其实现语句为：

```
t= linspace(-6,6);                          % 线性划分区间
x=sinc(t);
plot(t,x);
```

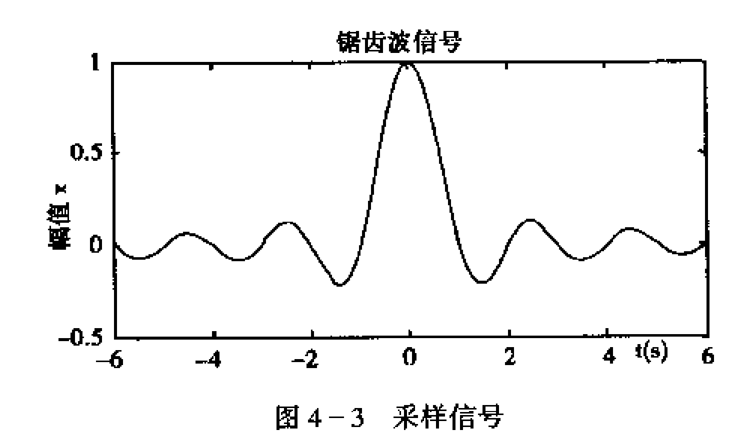


图4-3　采样信号

3. chirp信号

函数chirp()能够产生一种扫射频率信号，其特点是信号的瞬时频率随时间的推进按照一定的规律变化，如图4-4所示。其实现语句如下：

```
t=0:1/500:3;                      % 抽样频率为1KHz,抽样时间为3秒
x=chirp(t,0,2,400);               % 0时刻为Dc信号,1秒时频率为200Hz
specgram(x,256,1000,256,250);     % 画出频谱图
```

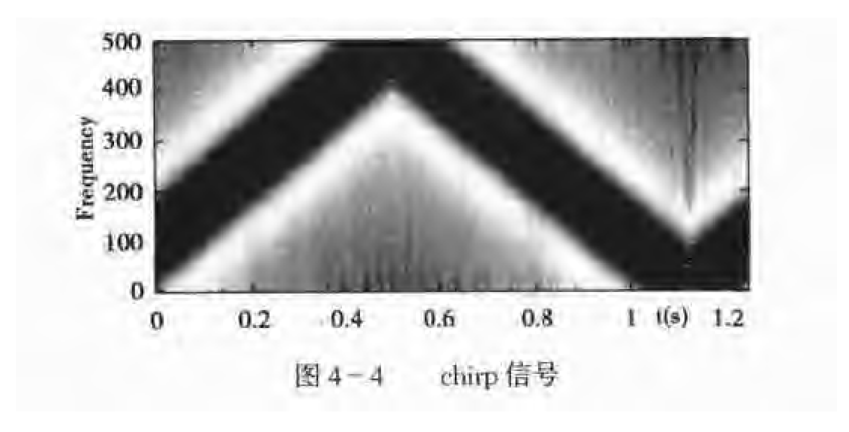


图4-4　chirp信号

提示：Simulink子模块库的Source子模块集中的Chirp Signal模块也可产生chirp信号。

4. Dirichlet(狄利克雷)信号

Dirichlet(狄利克雷)信号，也称周期sinc信号，其数学定义式为：

$$\text{diric}(x)=\begin{cases}(-1)^{k(n-1)} & x=2k\pi, k=0,\pm 1,\cdots \\ \dfrac{\sin(nx/2)}{n\sin(x/2)} & \text{其他}\end{cases} \tag{4-2}$$

以下的 MATLAB 语句,用于画出 n=7 和 n=8 时的 dirichlet 信号,如图 4-5 所示。

```
x = linspace(0,4 * pi,300);
plot(x,diric(x,7));
figure;
plot(x,diric(x,8));
```

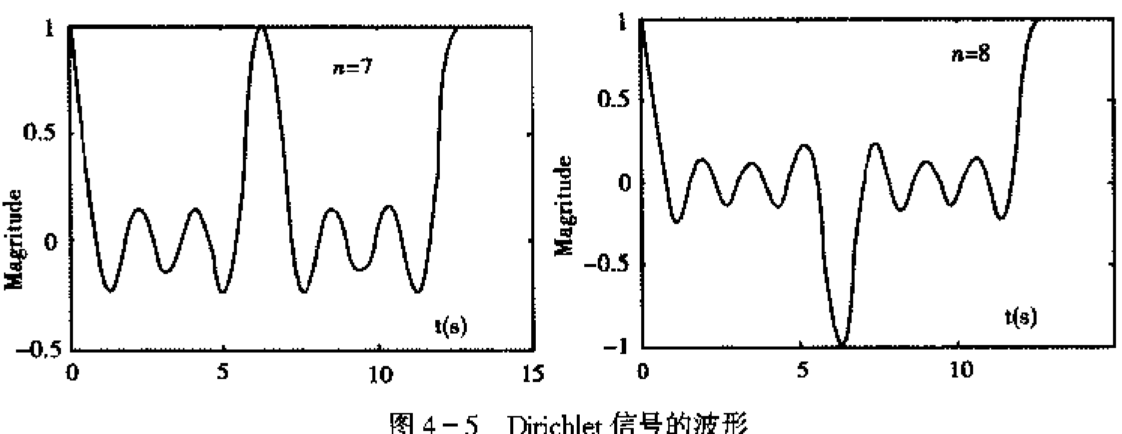


图 4-5 Dirichlet 信号的波形

5. 脉冲序列信号

脉冲序列信号也称脉冲调制信号,函数 pulstran()能产生脉冲调制信号,如图 4-6 所示。MATLAB 实现语句为:

```
T = 0:1/50E3:10E-3;                          % 抽样频率为 50KHz,抽样时间为 10ms
D = [0:1/1E3:10E-3;0.8.^(0:10)]';            % 第一列说明每一个脉冲的延迟时间,第二列
                                             % 说明一个脉冲的幅值衰减
Y = pulstran(T,D,'gauspuls',10E3,0.5);       % 调制波为高斯噪声,频率为 1KHz,带宽为 50%
plot(T,Y);
```

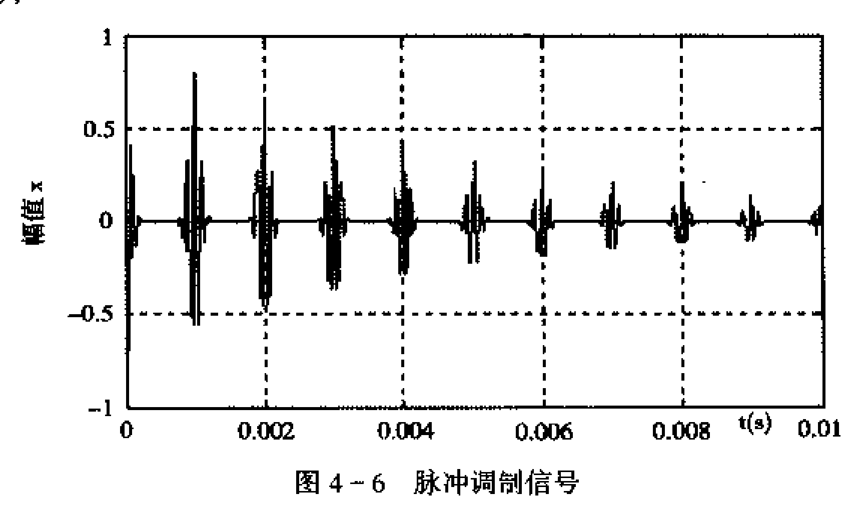


图 4-6 脉冲调制信号

另外,信号序列可进行一些操作运算,如加、乘、改变比例、折叠、抽样和、求信号能量、

求信号功率以及求两信号序列的卷积等，如表 4－2 所示。

表 4－2　信号序列之间的数学运算

信号序列的数学运算	描　　述
信号相加	$X(N)=X_1(N)+X_2(N)$
信号相乘	$X(N)=X_1(N)^T X_2(N)$
信号数乘	$X(N)=K*Y(N)$
信号位移	$X(N)=Y(N-K)$
信号折叠	$X(N)=Y(-N)$
信号样本和	$X=\sum_{n=n1}^{n2} y(n)$
信号样本积	$x=\prod_{n=n1}^{n2} y(n)$
信号的能量	$E_x=\sum_{n=n1}^{n2} \|x(n)\|^2$
信号的功率	$P_x=\frac{1}{n_2-n_1}\sum_{n=n1}^{n2} \|x(n)\|^2$

4.1.3　离散时间系统

离散时间系统可以用图 4－7 所示的结构来描述：

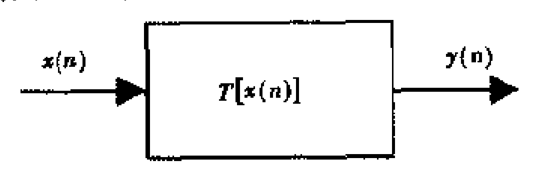


图 4－7　离散时间系统

在图 4－7 中，$x(n)$为输入序列，通常称为系统的激励；$y(n)$为输出序列，通常称为系统的响应。$T[x(n)]$描述了一个离散时间系统，该系统将一个输入序列变换成另外一个序列输出。

目前人们主要研究的是因果稳定线性移不变系统。下面对因果稳定线性移不变系统进行详细解释。

1. 因果系统

因果系统是指某时刻的输出取决于此时刻和此时刻的信号输入的系统，即 $n=n_0$ 时的输出 $y(n_0)$取决于 $n<n_0$ 的输入 $x(n)$。对于因果系统，如果 $n<n_0$ 时，$x_1(n)=$

$x_2(n)$,则 $n<n_0$ 的输出 $y_1(n)=y_2(n)$。只有因果系统才是物理上可实现的系统。

线性移不变系统成为因果系统的充分必要条件是:

$$h(n)=0, n<0 \tag{4-3}$$

2. 稳定系统

稳定系统是指由有界输入产生有界输出(BIBO)的系统。线性移不变系统成为稳定系统的充分必要条件是:

$$\sum_{n=-\infty}^{+\infty} |h(n)| = p < +\infty \tag{4-4}$$

即单位抽样响应绝对可和。

3. 线性系统

满足叠加原理的系统称为线性系统,对于线性系统,若输入为 $x_1(n)$和 $x_2(n)$,输出分别为 $y_1(n)$和 $y_2(n)$,即:

$$y_1(n)=T[x_1(n)], \qquad y_2(n)=T[x_2(n)] \tag{4-5}$$

则当输入为 $ax_1(n)+bx_2(n)$时,输出为 $ay_1(n)+by_2(n)$。其中 a、b 为任意常数,即:

$$T[ax_1(n)+bx_2(n)]= aT[x_1(n)]+bT[x_2(n)]= ay_1(n)+by_2(n) \tag{4-6}$$

4. 移不变系统

若系统响应与激励系统的时刻无关,则称系统为移不变系统。即若输入 $x(n)$产生输出为 $y(n)$,则输入 $x(n-m)$产生输出 $y(n-m)$,也就是说输入移动 m 位,其输出也相应移动 m 位,而幅值保持不变。

对于移不变系统,若:

$$T[x(n)]=y(n) \tag{4-7}$$

则

$$T[x(n-m)]=y(n-m) \tag{4-8}$$

其中 m 为任意整数。

连续时间线性移不变系统的输入输出关系常用常微分方程表示,而离散时间线性移不变系统的输入输出关系常用以下形式的常系数线性差分方程表示:

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{m=0}^{M} b_m x(n-m) \tag{4-9}$$

其中系数 $a_1,a_2,\ldots,a_N;b_1,b_2,\ldots,b_M$ 为常数。差分方程的阶数等于未知序列(指 $y(n)$)的最高值与最低值之差,上式为 N 阶差分方程。

求解常系数线性差分方程既可以用序列域求解法,也可以用变换域求解法,后一种方法将在本章以后的内容进行介绍,前一种方法主要是卷积和计算方法,即:

$$y(n) = T[x(n)] = x(n) * h(n) = \sum_{k=-\infty}^{+\infty} x(k)h(n-k) \qquad (4-10)$$

线性移不变系统可用它的单位抽样响应 $h(n)$ 来表征，即：

$$h(n) = T[\delta(n)] \qquad (4-11)$$

注意：conv()函数假定两个序列 x 和 h，都是从 n=0 开始的，若序列从某一负值开始，则不能直接调用 conv()函数，如下例所示。

例 4-1　已知离散系统的输入和冲激响应为：

$$x(n) = [1,4,3,5,1,2,3,5]$$

$$h(n) = [4,2,4,0,4,2]$$

求出系统的响应，并绘制出系统的响应图。

当求卷积 $y(n)=x(n)*h(n)$ 时，可直接调用 MATLAB 中的 conv()函数。其格式为：

```
y=conv(x,h);                          % x,h为参与卷积的两个序列
```

程序代码如下：

```
%   MATLAB program4-1
%   Solve Convolution problem
x=[1 4 3 5 1 2 3 5];                  % 输入两个序列
nx = -4:3;
h=[4 2 4 0 4 2];
nh=-3:2;
y =conv(x,h);                         %   求卷积
ny1=nx(1)+nh(1);
ny2=nx(length(nx))+nh(length(nh));
ny=[ny1:ny2];
subplot(3,1,1);
stem(nx,x);
title('输入');
subplot(3,1,2);
stem(nh,h);
title('冲激响应');
subplot(3,1,3);
stem(ny,y);
title('输出');
```

程序运行的结果为：

```
y=
    4  18  24  42  30  48  40  60  36  30  16  26  10
ny=
    -7  -6  -5  -4  -3  -2  -1   0   1   2   3   4   5
```

系统的响应图如图 4-8 所示。

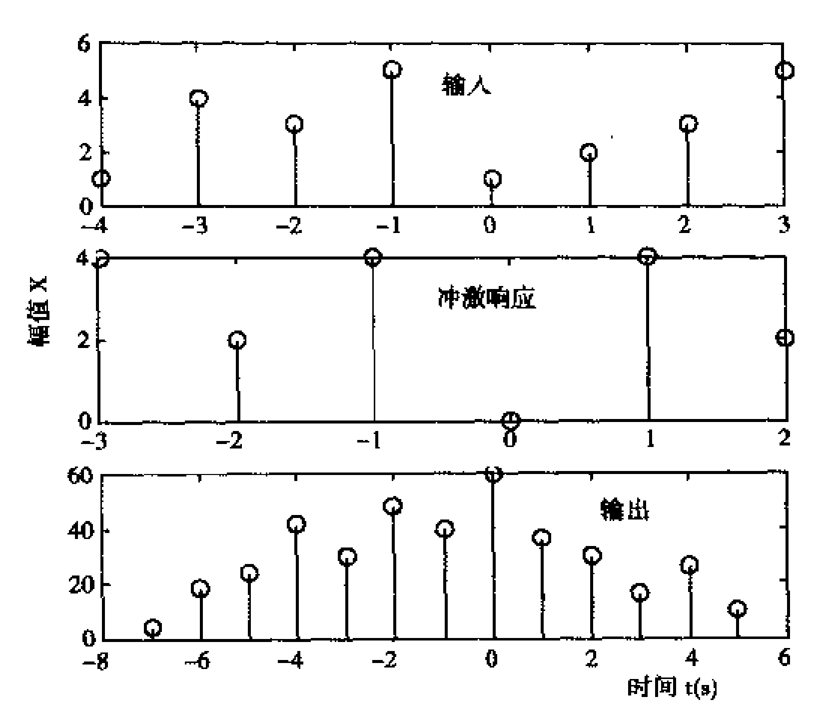


图 4-8　例 4-1 中系统的冲激响应

4.2　信号的变换域分析

4.2.1　Z 变换分析

在离散时间信号与系统中,变换域分析法主要是 Z 变换法和傅立叶变换法。Z 变换把描述离散系统的差分方程转化为简单的代数方程,使其求解大为简化。因此,对求解离散时间系统而言,Z 变换是一个极其重要的数学工具。另外,Z 变换在信号处理和系统动态特性研究中起着重要作用。下面先来讨论 Z 变换的定义和性质,再讨论 Z 变换的 MATLAB 实现及其应用。

1. Z 变换定义及收敛域

若信号序列为 $x(n)$,则幂级数表示为:

$$x(z) = \sum_{n=-\infty}^{+\infty} x(n) z^{-n} \qquad (4-12)$$

式(4-12)称为序列 $x(n)$的 Z 变换,其中 z 为变量,亦可将 $x(n)$的 Z 变换表示为:

$$Z[x(n)] = x(z) \qquad (4-13)$$

对于任意给定的序列 x(n),使 x(z)收敛的所有 Z 值集合称为 Z 变换的收敛域。即满足:

$$\sum_{n=-\infty}^{+\infty} |x(n) z^{-n}| = M < +\infty \qquad (4-14)$$

其中,|Z|的范围即为 Z 变换收敛域。接下来分别讨论不同形式的序列的收敛域。

有限长序列及收敛域

$$x(n)=\begin{cases}x(n) & n_1\leqslant n\leqslant n_2\\ 0 & \text{其他}\end{cases} \tag{4-15}$$

收敛域为 $0<|Z|<+\infty$,即除 $z=0$ 和 $z=+\infty$ 以外的开域 $(0,+\infty)$。特殊情况如下,当 $n_1\geqslant 0$,收敛域为 $0<|z|\leqslant+\infty$;当 $n_2\leqslant 0$,收敛域为 $0\leqslant|z|<+\infty$。

右边序列及收敛域

$$x(n)=\begin{cases}x(n) & n\geqslant n_1\\ 0 & n<n_1\end{cases} \tag{4-16}$$

收敛域为 $R_{x-}<|Z|<+\infty$,即圆外区域。

左边序列及收敛域

$$x(n)=\begin{cases}x(n) & n\leqslant n_2\\ 0 & n>n_2\end{cases} \tag{4-17}$$

收敛域为 $0<|Z|<R_{x+}$,即圆内区域。

双边序列及收敛域

双边序列 n 为任意值时 $x(n)$ 皆有值。其收敛域为:$R_{x-}<|Z|<R_{x+}$。$|Z|=1$ 称为 Z 平面上的单位圆,当收敛域包括单位圆时,则在单位圆上计算 $X(z)$,它实际上是傅立叶变换 $X(e^{j})$:

$$X(Z)\Big|_{z=e^{jw}}=x(e^{jw})=\sum_{n=-\infty}^{+\infty}x(n)e^{-jwn}=F[x(n)] \tag{4-18}$$

因此,傅立叶变换(DFT)可以被视为 Z 变换的特殊情况。

2. Z 变换的性质

Z 变换的性质如表 4-3 所示。

表 4-3　Z 变换性质

性质	序列 Z 变换	收敛域
线性	$Z[X(N)]=X(Z)$ $Z[Y(N)]=Y(Z)$ $Z[AX(N)+BY(N)]=AX(Z)+BY(Z)$	$R_{x-}<\|Z\|<R_{x+}$ $R_{y-}<\|Z\|<R_{y+}$ $\max(R_{x-},R_{y-})<\|Z\|<\min(R_{x+},R_{y+})$
序列位移	$Z[X(N)]=X(Z)$ $Z[X(N-M)]=Z^{-M}X(Z)$	$R_{x-}<\|Z\|<R_{x+}$ $R_{x-}<\|Z\|<R_{x+}$
频率位移	$Z[X(N)]=X(Z)$ $Z[A^NX(N)]=X(Z/A)$	$R_{x-}<\|Z\|<R_{x+}$ $\|a\|R_{x-}<\|Z\|<\|a\|R_{x+}$

（续表）

性质	序列 Z 变换	收敛域
Z 域微分	$Z[X(N)]=X(Z)$ $Z[NX(N)]=-Z*(DX(Z)/DZ)$	$R_{x-}<\|Z\|<R_{x+}$ $R_{x-}<\|Z\|<R_{x+}$
共轭序列	$Z[X(N)]=X(Z)$ $Z[X*(N)]=X*(Z)$	$R_{x-}<\|Z\|<R_{x+}$ $R_{x}\ <\|Z\|<R_{x+}$
翻褶序列	$Z[X(N)]=X(Z)$ $Z[X(-N)]=X(1/Z)$	$R_{x-}<\|Z\|<R_{x+}$ $1/R_{x-}<\|Z\|<1/R_{x-}$
有限项累加特性	$Z[X(N)]=X(Z)$ $Z[\sum_{m=0}^{n}x(m)]=\frac{z}{z-1}X(z)$	$\|Z\|>R_{x-}$ $\|Z\|>\max(R_{x-},1)$
序列卷积	$Z[X(N)]=X(Z)$ $Z[H(N)]=H(Z)$ $Y(Z)=Z[Y(N)]=Z[X(N)*H(N)]=$ $H(Z)*X(Z)$	$R_{x-}<\|Z\|<R_{x+}$ $R_{h-}<\|Z\|<R_{h+}$ $\max(R_{x-},R_{h-})<\|Z\|<\min(R_{x+},R_{h+})$

3. Z 域系统表示

将系统函数 $H(z)$ 定义为：

$$H(z)=Z[h(n)]=\sum_{n=-\infty}^{+\infty}h(n)z^{-n}\qquad R_{h-}<|z|<R_{h+}\tag{4-19}$$

用系统函数 $H(z)$ 和卷积性质，可得到系统输入 $X(z)$ 与输出 $Y(z)$ 之间的关系：

$$Y(z)=H(z)X(z)\tag{4-20}$$

将系统用差分方程表示为：

$$y(n)+\sum_{k=1}^{N}a_k y(n-k)=\sum_{m=0}^{M}b_m x(n-m)\tag{4-21}$$

由差分方程可以直接写出函数 $H(z)$：

$$H(z)=\frac{Y(z)}{X(z)}=\frac{\sum_{m=0}^{M}b_m z^{-m}}{1+\sum_{k=1}^{N}a_k z^{-k}}=b_0 e^{j(N-M)w}\frac{\prod_{i=1}^{m}(z-Z_i)}{\prod_{k=1}^{N}(z-P_k)}\tag{4-22}$$

当 $H(z)$ 的收敛域包含单位圆时，可以计算出单位圆上的 $H(z)$。令(4-22)式中

$z=e^{jw}$,将得到频域传递函数 $H(e^{jw})$:

$$H(e^{jw}) = b_0 e^{j(N-M)w} \frac{\prod_{i=1}^{M}(e^{jw} - Z_i)}{\prod_{k=1}^{N}(e^{jw} - P_k)} \tag{4-23}$$

MATLAB工具箱提供了函数 freqz(),通过它可以直接给出传递函数形式表示的幅频特性和相频特性曲线。

例4-2 一个因果线性移不变系统

$$y(n)-0.4y(n-1)+0.8y(n-2)=x(n)+0.7x(n-1)$$

(1)求 $H(z)$,并给出冲激响应曲线 $h(n)$和阶跃响应曲线 $u(n)$;

(2)求 $H(e^{jw})$,并给出幅频特性曲线和相频特性曲线。

解:$H(z)$可以由差分方程直接求出:

$$H(z)=\frac{1+0.7z^{-1}}{1-0.4z^{-1}+0.8z^{-2}}$$

于是:

$$H(e^{jw})=\frac{1+0.7e^{-jw}}{1-0.4e^{-jw}+0.8e^{-j2w}}$$

执行下面的MATLAB程序可以得到解:

```
%MATLAB program4-2
%Transfer, Step & Impulse response
clear all; clc;
a=[1 -0.4 0.8]; b=[1 0.7];                 % 输入传递函数的系数
figure(1);                                 % 绘制冲激响应曲线
subplot(2,1,1);
dimpulse(b,a,20);
title('冲激响应');
xlabel('n');
ylabel('h(n)');
subplot(2,1,2);                            % 绘制阶跃响应曲线
dstep(b,a,50);
title('阶跃响应');
xlabel('n');
ylabel('h(n)');
figure(2);                                 % 绘制幅频特性和相频特性曲线
t=[0:1:500]*pi/500;freqz(b,a,t);
```

系统冲激响应 $h(n)$曲线和阶跃响应曲线 $u(n)$,如图4-9所示;幅频特性曲线和相频特性曲线如图4-10所示。

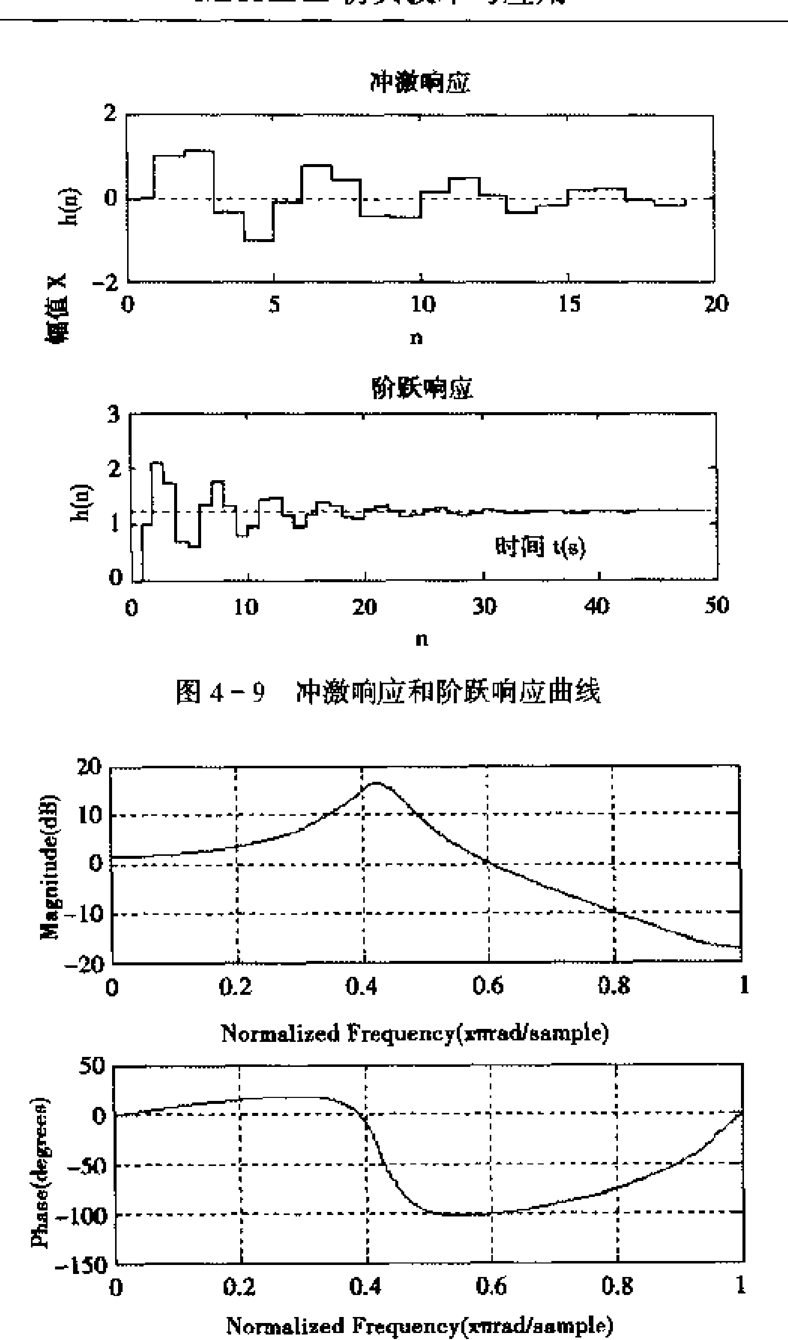


图 4－9　冲激响应和阶跃响应曲线

图 4－10　幅频特性和相频特性曲线

4．离散系统的差分方程

线性移不变系统可以用以下形式的线性常系数差分方程描述：

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{m=0}^{M} b_m x(n-m) \tag{4-24}$$

如果 $a_N \neq 0$，则此差分方程是 N 阶。通常，可将式(4－24)写成如下的形式：

$$y(n) = \sum_{m=0}^{M} b_m x(n-m) - \sum_{k=1}^{N} a_k y(n-k) \tag{4-25}$$

该差分方程的解为：

$$y(n) = yH(n) + yP(n) \tag{4-26}$$

差分方程的解，可以表示成通解和特解之和，也可以表示成暂态响应和稳态响应之和，还可以表示成为零状态响应和零输入响应之和。

MATLAB 提供了函数 filtic()求差分方程的零状态响应，提供了函数 filter()求差分方程的完全解。

例 4-3　*求解差分方程*

$(y)n = x(n) + x(n-1) + x(n-2) + 0.95y(n-1) - 0.9025y(n-2), n \geqslant 0$

其中，$x(n) = \cos(\frac{\pi n}{6}), y(-1) = -2, y(-2) = -4, x(-1) = x(-2) = 2$。

将差分方程写成以下标准形式：

$y(n) - 0.95y(n-1) + 0.9025y(n-2) = [x(n) + x(n-1) + x(n-2)]$

执行下面的 MATLAB 程序，即可得到差分方程的解。

```
%MATLAB program4-3
%Solve difference equation
clear all;clc;
b=[1 1 1];
a=[1,-0.95,0.9025];
Y=[-2,-4];                    % 输入初始条件
X=[1,2];
xic=filtic(b,a,Y,X)           % filter()函数选择初始条件
n=[0:50];
x=cos(n*pi/6);
y=filter(b,a,x,xic);
plot(n,y);
```

计算结果：xic = 4.7100　　2.8050

系统的完全响应曲线如图 4-11 所示。

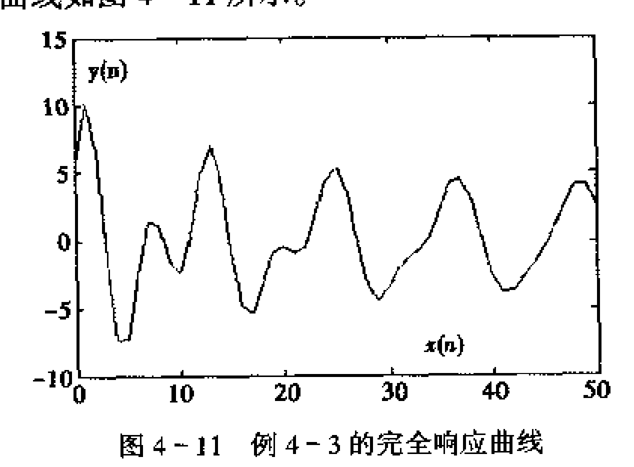


图 4-11　例 4-3 的完全响应曲线

5. Z 变换及其反变换的 MATLAB 实现

离散因果序列的 Z 变换及其反变换的定义如下：

Z 变换：$F(Z) = \sum_{n=0}^{+\infty} f(n) z^{-n}$ (4-27)

Z 反变换：$f(n) = Z^{-1}\{F(z)\}$ (4-28)

最常见的求解频率域序列的 Z 反变换表达式的具体方法有三种，即：幂级数展开法、部分分式法和围线积分法。MATLAB 的信号处理工具箱采用围线积分法设计了求取 Z 反变换的 iztrans 指令，相应的数学表达式如下：

$$f(n) = \frac{1}{2\pi j}\oint_{\Gamma} F(z) z^{n-1} dz \qquad (4-29)$$

Z 变换和 Z 反变换的具体指令分别为：

```
Fz = ztrans(fn,n,z);                    % 求时域序列 fn 的 Z 变换 Fz
fn = iztrans(Fz,n,z);                   % 求频域序列 Fz 的 Z 反变换 fn
```

例 4-4 求序列 $f(n)=\begin{cases} 0 & n<0 \\ 4 & n=0 \\ 5(1-0.5^n) & n>0 \end{cases}$ 的 Z 变换，并用反变换验证。

MATLAB 程序代码如下：

```
%MATLAB program 4-4
% Z transform and Z inverse transform
clear all;
clc;
syms n;
syms z;
delta = sym("charfcn[0](n)");
d0 = subs(delta,n,0);
d15 = subs(delta,n,15);
fn = 4 * delta + 5 * (1 - 0.5^n);
fz = ztrans(fn,n,z)
Fz_n = iztrans(fz,z,n)
```

程序执行结果如下：

```
fz =
      4 + 5 * z/(z - 1) - 10 * z/(2 * z - 1)
Fz_n =
      4 * charfcn[0](n) + 5 - 5 * (1/2)^n
```

4.2.2 Fourier(傅立叶)变换分析

在数字信号处理中，有限长序列是一种非常重要的序列，而离散傅立叶变换(DFT)则是研究有限长序列的一种重要工具。由于 DFT 存在着一种快速而有效的算法——快速傅立叶变换(FFT)，因而 DFT 在各种数字信号处理算法中起着核心作用。

1. DFT 定义

DFT 的定义如下：

正变换：$X(k) = \mathrm{DFT}[x(n)] = \sum_{n=0}^{N-1} x(n)\mathrm{e}^{-j\frac{2\pi T}{N}nk}, \quad 0 \leqslant k \leqslant N-1$ (4－30)

反变换：$x(n) = \mathrm{IDFT}[x(k)] = \frac{1}{N}\sum_{k=0}^{N-1} x(k)^{j\frac{2\pi}{N}nk}, \quad 0 \leqslant n \leqslant N-1$ (4－31)

2. DFT 性质

设两个序列 $x_1(n)$和 $x_2(n)$都是 N 点有限长序列，且：

$$\mathrm{DFT}[x_1(n)] = x_1(k), \qquad \mathrm{DFT}[x_2(n)] = x_2(k) \tag{4-32}$$

则 DFT 具有以下性质：

1)线性性质

$$\mathrm{DFT}[a\,x_1(n) + bx_2(n)] = a\,x_1(k) + b\,x_2(k), \quad a, b \text{ 是任意常数} \tag{4-33}$$

2)序列圆周移位

一个有限长序列 $x(n)$的圆周移位定义为：

$$x_m(n) = x((n+m))_N R_N(n) \tag{4-34}$$

其中，$x((n+m))_N$ 表示$x(n)$的圆周延拓序列 $\overline{x}(n)$的移位。

$$x((n+m))_N = \overline{x}(n+m) \tag{4-35}$$

有限长序列圆周移位后的 DFT 为：

$$x_m(k) = \mathrm{DFT}[x((x+m))_N R_N(n)] = W_N^{-mk} x(k) \tag{4-36}$$

这里 $R_N(n)$是矩形波信号，即：

$$R_N(n) = \begin{cases} 1 & 0 \leqslant n \leqslant N-1 \\ 0 & \text{其他} \end{cases}, \qquad W_N = e^{j\frac{2\pi}{N}} \tag{4-37}$$

3)共轭对称性

设 $\mathrm{DFT}[x(n)] = \mathrm{DFT}\{\mathrm{Re}[x(n)] + j\,\mathrm{Im}[x(n)]\}$，则有：

$$\begin{aligned} &\mathrm{DFT}[x^*(n)] = x^*((-k))_N R_N(k) = x^*((N-k))_N R_N(k) \\ &\mathrm{DFT}[x^*((-n))_N R_N(n)] = x^*(k) \\ &\mathrm{DFT}\{\mathrm{Re}[x(n)]\} = \frac{1}{2}[x((k))_N + x^*((N-k))_N] R_N(k) \\ &\mathrm{DFT}\{j\,\mathrm{Im}[x(n)]\} = \frac{1}{2}[x((k))_N - x^*((N-k))_N] R_N(k) \end{aligned} \tag{4-38}$$

4)圆周卷积和

若 $Y(k) = x_1(k)x_2(k)$则：

$$y(n) = \mathrm{IDFT}[Y(k)] = [\sum_{m=0}^{N-1} x_1(m)x_2((n-m))_N] R_N(n) = x_1(n) \otimes x_2(n) \tag{4-39}$$

5)序列乘积

DFT 序列乘积的性质如下：

$$\mathrm{DFT}[x_1(n)x_2(n)]=\frac{1}{N}x_1(k)\otimes x_2(k) \tag{4-40}$$

3. FFT(快速傅立叶变换)

N 点序列 $x(n)$ 的 N 点 DFT 可以表示成:

$$x(k)=\sum_{n=0}^{N-1}x(n)W_N^{nk},0\leqslant k\leqslant N-1 \tag{4-41}$$

其中,$W_N=\mathrm{e}^{-j\frac{2\pi}{N}}$

利用系数 $\{W_N^{nk}\}$ 的周期性:

$$W_N^{nk}=W^{n+N}{k_N}=W_N^{k+N} \tag{4-42}$$

DFT 运算中的某些项可以合并,利用对称性和周期性可以将长序列的 DFT 分解为短序列的 DFT。其对称性为:

$$W_N^{nk+\frac{N}{2}}=-W_N^{nk} \tag{4-43}$$

FFT 正是基于这个思路发展起来的,FFT 算法基本上可以分成两大类:按时间抽取(Decimation－In－Time, DIT)法和按频率抽取(Decimation－In－Frequency, DIF)法。

DFT 公式和 IDFT 公式分别如下:

$$\begin{aligned}&\text{DFT 公式: } x(k)=\sum_{n=0}^{N-1}x(n)W_N^{nk},0\leqslant k\leqslant N-1\\&\text{IDFT 公式: } x(n)=\frac{1}{N}\sum_{k=0}^{N-1}x(k)W_N^{-nk},0\leqslant n\leqslant N-1\end{aligned} \tag{4-44}$$

可以看出,只要把 DFT 运算中的每一个系数 W_N^{nk} 换成 W_N^{-nk},并且乘以常数$\frac{1}{N}$,就可以用于 IDFT 运算。

在 MATLAB 中,可以直接利用 fft() 函数进行 FFT 运算,用 ifft()函数进行反 FFT 运算。MATLAB 根据以上的定义给出的利用 FFT 算法实现 DFT 和 IDFT 的指令如下:

```
Xf=fft(Xt,N,Dim);          % 计算 N 点 Xt 序列的 N 点 DFTXf
Xt=ifft(Xf,N,Dim);         % 计算 N 点 Xf 序列的 N 点离散傅立叶反变换 Xt
```

FFT 技术是求解离散数据傅立叶变换最实用也是最通用的方法。MATLAB 提供了内在函数 fft(),通过它可以有效地求解 FFT 问题,该函数的另外一个显著特点是它可以对任意长度的向量进行变换。此外,MATLAB 还提供了其他一些数字信号分析函数,这些函数允许用户完成很多信号处理任务。这些函数如表 4－4 所示。

提示: 在 MATLAB 的数字信号处理子模块库(DSP Blockset)的变换子模块集(Transforms)中提供了用于快速傅立叶变换及其反变换、对幅值进行傅立叶变换的仿真模块,如图 4－12 所示。

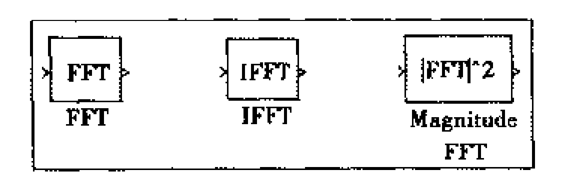


图 4－12　傅立叶变换的仿真模块

表 4－4　MATLAB 提供的数字信号分析函数

函　数	作　用
FFT(X)	进行向量 X 的 DFT。如果 X 的长度是 2 的幂,则用 FFT 算法
FFT(X,N)	得到一个长度为 N 元素是 X 中前 N 个元素 DFT 值的向量
FFT(A)	求矩阵 A 的列 DFT 矩阵
FFT(A,N,DIM)	求多维数组 A 中 DIM 维内列 DFT 矩阵
IFFT(X)	求向量 X 的离散傅立叶反变换
FFT2(A)	求矩阵 A 的二维 DFT 矩阵。如果 A＝A 是一个向量,则这个命令等于 FFT(A)命令
FFT2(A,M,N)	求矩阵 A 中相应元素的二维 DFT 矩阵。这个矩阵大小为 M＊N
IFFT2(A)	求矩阵 A 的二维离散傅立叶反变换矩阵
FFTN(A,SIZE)	在给定了数组的大小 SIZE 的情况下,求 N 维数组 A 的 N 维 DFT 数组
IFFTN(A,SIZE)	在给定了数组的大小 SIZE 的情况下,求 N 维数组 A 的 N 维离散傅立叶反变换数组
FFTSHIFT(A)	返回一个将矩阵 A 的第 1 象限和第 3 象限、第 2 象限和第 4 象限互换的数组
IFFTSHIFT(A)	FFTSHIFT(A)命令的反变换
FILTER(B,A,X)	由向量 A 和 B 所描述形成的滤波器对向量 X 进行数字滤波,产生滤波后的数据
Y＝FILTER2(H,X)	用在矩阵 H 中的 FIR 滤波器处理 X 中的数据。结果 Y 由二维卷积计算得到,它包含卷积的中心部分,且与 X 的大小相同
Y＝FILTER2(H,X,FORM)	Y 由二维卷积计算得到,其维数由参数 FORM 规定

4. 应用实例

例 4－5　用 FFT 对信号进行谱分析,其中信号为:$x(t)=\cos(8\pi t)+\cos(16\pi t)+\cos(20\pi)t$,选择抽样频率 Fs＝64Hz,抽样点数分别取为 16、32 和 64。

MATLAB 程序如下:

```
%MATLAB program 4-5
%Spectra analysis
clear all;
clc;
Fs=64;
N=64;
```

```
t=1/Fs;
f1=4;f2=8;f3=10;
n=0:N-1;
x=cos(2*pi*t*f1*n)+cos(2*pi*t*f2*n)+cos(2*pi*t*f3*n);
stem(x);
y=t*abs(fft(x,N));
figure;
stem(y,'*');
```

程序运行后，得到原信号和处理后信号的频谱图，如图4－13和4－14所示。

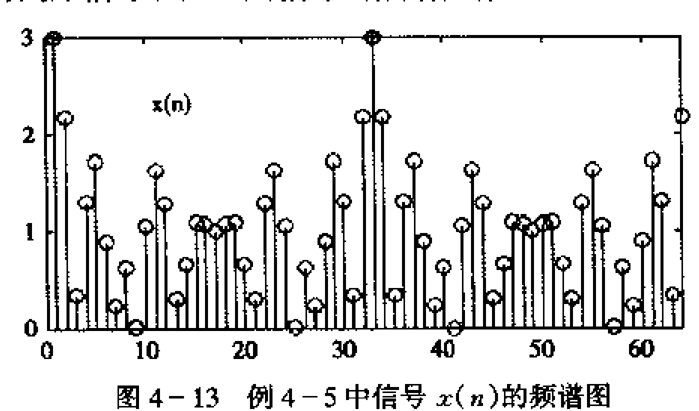


图4－13　例4－5中信号 $x(n)$ 的频谱图

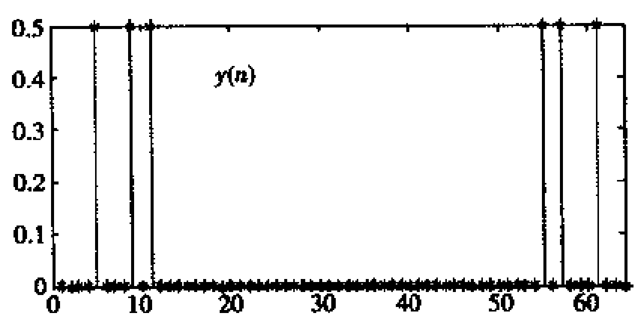


图4－14　例4－5中 $y(n)$ 的频谱图

例4－6　利用FFT对有干扰的信号进行处理，其中信号为：$x(t)=\sin(50\pi t)+\sin(100\pi t)$　$(0.1ms\leqslant t\leqslant 0.25ms)$，干扰信号为：$y(t)=x+2*randn(size(t))$。

对有干扰的信号，通常采用滤波的方法对其进行处理。

MATLAB程序如下：

```
%MATLAB program 4-6
%Digtal sognal processing
clear all; clc; t=0:0.001:0.25;
x=sin(2*pi*25*t)+sin(2*pi*50*t);
y=x+2*randn(size(t));
plot(y(1:50));
title('Noisy Time Domain Signal');
Y=fft(y,256);
py=Y.*conj(Y)/256;
f=1000/256*(0:127);
```

```
figure;  plot(f,py(1:128));
title('Power Spectral Density');
xlabel('Frequency (Hz)');
figure;  plot(f(1:50),py(1:50));
title('Power Spectral Density');
xlabel('Frequency (Hz)');
```

运行以上程序得到的噪声信号、有干扰信号和滤掉干扰信号后的频谱图分别如图4-15、4-16和4-17所示。

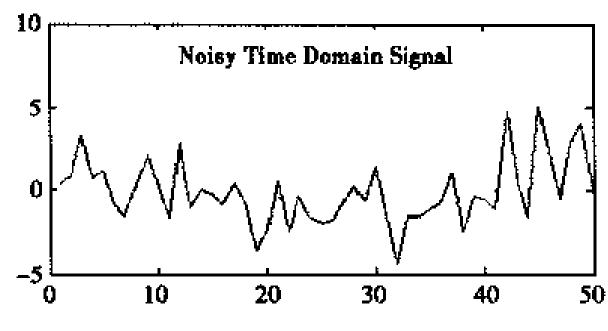


图4-15 例4-6中噪声信号的频谱图

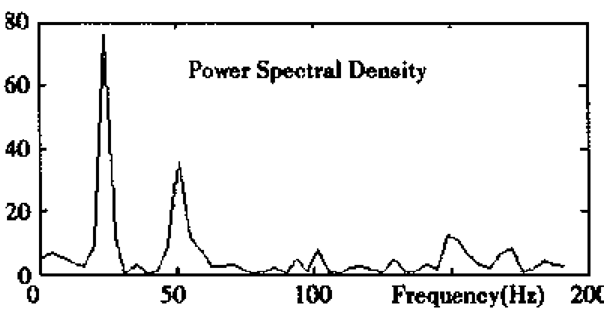


图4-16 例4-6中有干扰情况下信号的频谱图

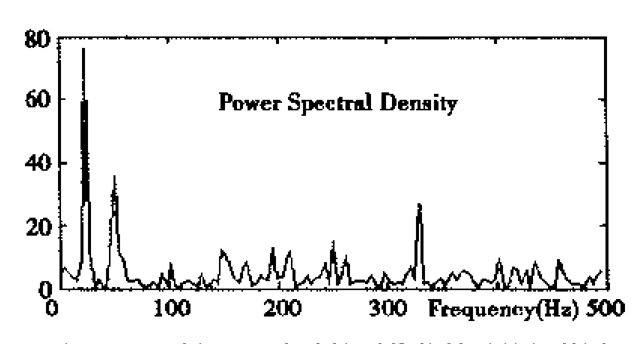


图4-17 例4-6中滤掉干扰信号后的频谱图

4.3 数字滤波器的设计

4.3.1 数字滤波器的结构

数字滤波器在数字信号处理中发挥着重要作用,它通过对采样数据信号进行数字运算处理来达到在频率域滤波的目的。对于数字滤波器,从实现方法上可以分为FIR数字滤波器和IIR数字滤波器,FIR(finite impulse response)滤波器是指由有限冲激响应所表示的数字滤波器;IIR(infinite impulse response)滤波器是指具有无限冲激响应的数字滤波器。下面分别介绍这两种滤波器的基本结构。

1. IIR滤波器的结构

一个IIR滤波器的系统函数为:

$$H(z)=\frac{Y(z)}{X(z)}=\frac{b_0+b_1z^{-1}+\cdots+b_Mz^{-m}}{1+a_1z^{-1}+\cdots+a_Nz^{-N}}=\frac{\sum_{m=0}^{M}b_mz^{-m}}{\sum_{n=0}^{N}a_nz^{-n}} \tag{4-45}$$

其中，a_n、b_m 是滤波器的系数，同时 $a_0=1$。如果 $a_N\neq0$，则(4－45)式所表示的滤波器的阶数是 N 阶。IIR 滤波器的差分方程表示为：

$$y(n)=\sum_{m=0}^{m}b_mx(n-m)-\sum_{m=0}^{N}a_my(n-m) \tag{4-46}$$

在工程应用中，通过四种结构来实现 IIR 滤波器：直接 I 型、直接 II 型、级联型和并联型。下面分别介绍这几种结构的 IIR 滤波器。

直接 I 型和直接 II 型

直接 I 型和直接 II 型结构统称为直接型结构。用直接型结构实现 IIR 滤波器，就是先将 IIR 滤波器的系统函数表示成(4－46)式的差分方程的标准型，然后用延迟元件、乘法器和加法器直接实现。

例 4－7　设有一个 5 阶 IIR 滤波器，其差分方程如下：

$$b_0x(n)+b_1x(n-1)+b_2x(n-2)+b_3x(n-3)+b_4x(n-4)+b_5x(n-5)=$$
$$y(n)+a_1y(n-1)+a_2y(n-2)+a_3y(n-3)+a_4y(n-4)+a_5y(n-5) \tag{4-47}$$

试用直接 I 型和直接 II 型结构分别实现。

解：先将差分方程(4－47)变换成标准型：

$$y(n)=b_0x(n)+b_1x(n-1)+b_2x(n-2)+b_3x(n-3)+b_4(n-4)+$$
$$b_5x(n-5)-a_1y(n-1)-a_2y(n-2)-a_3y(n-3)-a_4y(n-4)-a_5y(n-5)$$

于是可以直接画出它的直接 I 型结构，如图 4－18(a)所示。如果将直接 I 型结构中左右两部分交换一下，并把对应的延迟单元合并起来，则可以得到一种只需要较少延迟单元的简化结构，通常将其称之为直接 II 型结构或典型结构，如图 4－18(b)所示。

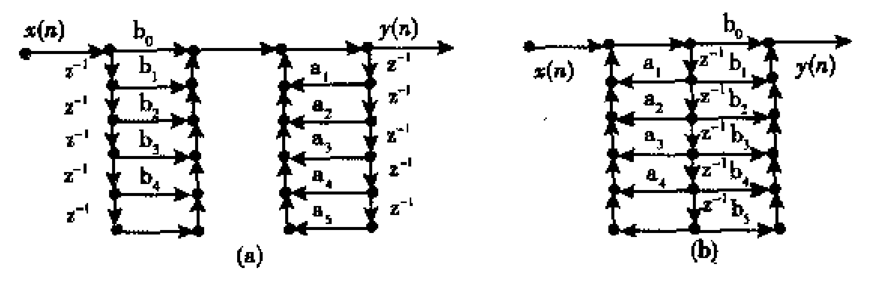


图 4－18　直接 I 型和直接 II 型结构

在 MATLAB 中，直接型结构用两个行向量描述，b 包含 $\{b_n\}$ 系数，a 包含 $\{a_n\}$ 系数。直接型结构可以由 MATLAB 提供的 filter()函数来实现。

级联型

在级联型中，系统函数需要分解成一阶或二阶数字滤波器传递函数的乘积。首先解出分子、分母多项式的特征值；然后把每一对共轭复根或任意两个实根组合在一起，得到二阶子系统。假设 N 为偶数，于是(4－45)式可以转化为下面的形式：

$$H(z)=\frac{b_0+b_1z^{-1}+\cdots+b_Mz^{-M}}{1+a_1z^{-1}+\cdots+a_Nz^{-N}}=b*\prod_{k=1}^{K}\frac{1+B_{k,1}z^{-1}+B_{k,2}z^{-2}}{1+A_{k,1}z^{-1}+A_{k,2}z^{-2}} \tag{4-48}$$

其中，$K=\frac{N}{2}$，$B_{k,1}$，$B_{k,2}$，$A_{k,1}$，$A_{k,2}$均为实数，表示二阶子系统的系数。二阶子系统如下所示：

$$H_k(z)=\frac{Y_{k+1}(z)}{Y_k(z)}=\frac{1+B_{k,1}z^{-1}+B_{k,2}z^{-2}}{1+A_{k,1}z^{-1}+a_{k,2}z^{-2}}\qquad k=1,2,\cdots,k \tag{4-49}$$

(4－49)式中的参量满足：

$$Y_1(z)=b_0x(z),\ Y_{k+1}(z)=Y(z) \tag{4-50}$$

在工程实际中，一般把(4－49)式所表示的结构称为双二阶节，它的输入是上一个双二阶节的输出，即第 K 个双二阶节的输出是第(K＋1)个双二阶节的输入，每一个双二阶节可用图 4－19 所示的直接 II 型结构实现，整个滤波器由双二阶节级联型实现。

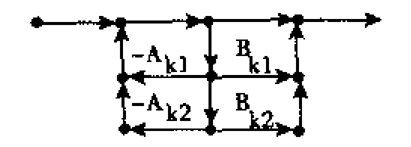


图 4－19　双二阶节结构

并联型

在并联型形式中，系统函数用部分分式展开式写成二阶子系统的和，形式如下：

$$H(z)=\frac{b_0+b_1z^{-1}+\cdots+b_Mz^{-M}}{1+a_1z^{-1}+\cdots+a_Nz^{-N}}=\sum_{k=1}^{K}\frac{B_{k,0}+B_{k,1}z^{-1}}{1+A_{k,1}z^{-1}+A_{k,2}z^{-2}}+\sum_{k=0}^{M-N}C_kz^{-k} \tag{4-51}$$

其中，$K=N/2$，$B_{k,1}$，$B_{k,2}$，$A_{k,1}$，$A_{k,2}$均为实数，表示二阶子系统的系数。而且只有当 M·N 才有后面的 FIR 部分，二阶子系统如下所示：

$$H_k(z)=\frac{Y_{k+1}(z)}{Y_k(z)}=\frac{B_{k,0}+B_{k,1}z^{-1}}{1+A_{k,1}z^{-1}+a_{k,2}z^{-2}},\quad k=0,1,\cdots,K \tag{4-52}$$

上式中的参量满足以下条件：

$$Y_k(z)=H_k(z)X(z);Y(z)=\sum Y_k(z);M<N \tag{4-53}$$

在工程实际中，一般把(4－52)式所示的结构称为有理双二阶节。滤波器的输入对所有双二阶节均有效，同时，若 M·N(FIR 部分)也是多项式的输入，这些环节和形成滤波器的输出。每一个有理双二阶节可以用如图 4－18(b)所示的直接 II 型结构实现。图4－20 给出了一个 4 阶 IIR 滤波器的并联型结构(M＝N＝4)。

2．FIR 滤波器结构

如果一个具有有限持续时间冲激响应的滤波器系统函数为：

$$H(z)=b_0+b_1z^{-1}+\cdots+b_{M-1}z^{-(M-1)}=\sum_{n=0}^{M-1}b_nz^{-n} \tag{4-54}$$

则其冲激响应为：

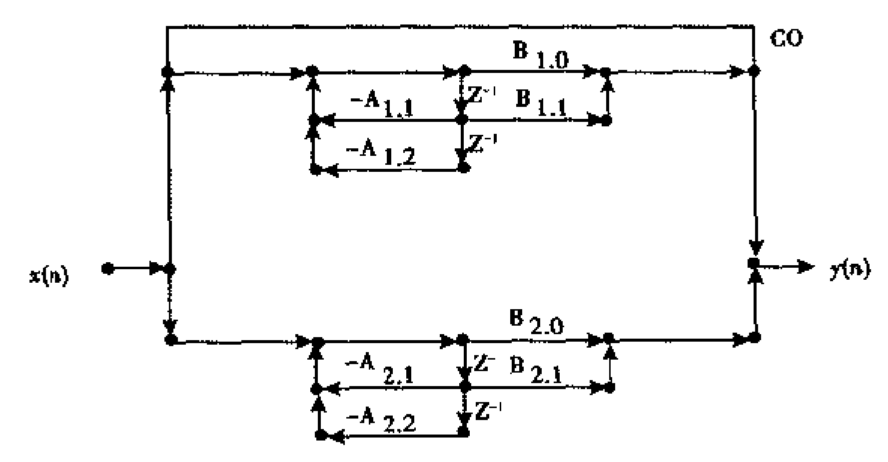


图 4－20　4 阶 IIR 滤波器的并联型结构

$$h(n)=\begin{cases} b_n & 0\leqslant n\leqslant M \\ 0 & 其他 \end{cases} \tag{4-55}$$

其差分方程可以描述为：

$$y(n)=b_0x(n)+b_1x(n-1)+\cdots+b_{M-1}x(n-M+1) \tag{4-56}$$

FIR 滤波器一般有 5 种结构：横截型、级联型、线性相位型、快速卷积型和频率采样型，由于后两种结构应用较少，故在本书只对前三种结构进行讨论。

横截型（直接型、卷积型）

横截型与 IIR 滤波器的直接型类似，只是没有反馈回路，因此它由抽头延迟线实现。下面以一个 3 阶滤波器（M＝4）为例进行介绍。

3 阶滤波器的差分方程描述为：

$$y(n)=b_0x(n)+b_1x(n-1)+b_2x(n-2)+b_3x(n-3)$$

它的横截型结构如图 4－21 所示，其转置结构如图 4－22 所示。

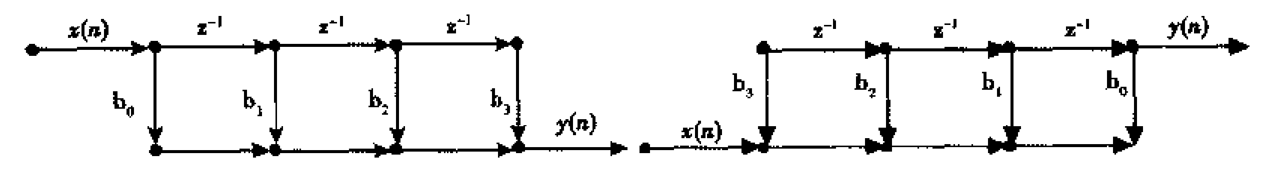


图 4－21　FIR 横截型结构　　　　图 4－22　FIR 横截型结构的转置结构

在 MATLAB 中，FIR 直接型结构由一个行向量描述，b 包含 $\{b_n\}$ 系数，向量 a 设置为 1。FIR 直接型结构可以由 MATLAB 提供的 filter()函数来实现。

级联型

级联型与 IIR 滤波器级联型类似，把系统函数 $H(z)$ 分解成二阶因子的乘积形式，二阶因子用横截型结构实现，整个滤波器用二阶因子的级联实现，从式（4－54）可以得到：

$$\begin{aligned} H(z) &= b_0+b_1z^{-1}+\cdots+b_{M-1}z^{-(M-1)} \\ &= b_0\left(1+\frac{b_1}{b_0}z^{-1}+\cdots+\frac{b_{M-1}}{b_0}z^{-(M-1)}\right) \\ &= b_0\prod_{K=0}^{K}(1+B_{k,1}z^{-1}+B_{k,2}z^{-2}) \end{aligned} \tag{4-57}$$

其中 $K=M/2$，$B_{k,1}z^{-1}$，$B_{k,2}$表示二阶子系统的系数，图4－23给出了滤波器 M＝5 时的结构图。

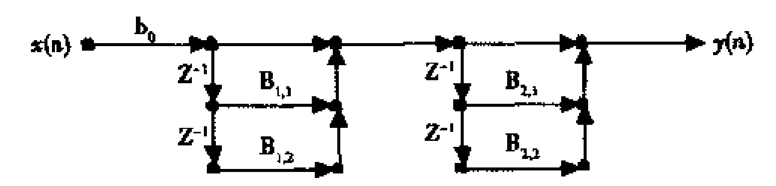


图4－23 FIR滤波器级联型结构

线性相位型

FIR滤波器的线性相位是非常重要的，因为数据以及图像处理都要求系统具有线性相位，而FIR滤波器由于它的冲激响应是有限长的，因而有可能做成严格线性相位的滤波器。即要求系统函数的相位为频率的线性函数，满足下列条件：

$$\angle H(e^{jw})=\beta-\alpha w,\ -\pi<\omega\leqslant\pi \tag{4-58}$$

其中 $\beta=0$ 或 $\pm\frac{\pi}{2}$，α 是常数，对于线性移不变因果性的FIR滤波器，它的冲激响应在区间$[0,M-1]$上，线性相位条件式(4－58)表明了 $h(n)$具有以下所示的对称性：

$$h(n)=h(M-1-n);\quad \beta=0,0\leqslant n\leqslant M-1 \tag{4-59}$$

$$h(n)=-h(M-1-n);\quad \beta=\pm\frac{\pi}{2},0\leqslant n\leqslant M-1 \tag{4-60}$$

满足(4－59)的冲激响应称为对称冲激响应，满足(4－60)的冲激响应称为反对称冲激响应，这些对称条件可以在称为线性相位的结构中使用。

差分方程式(4－56)具有式(4－59)中的对称冲激响应，即满足下式：

$$\begin{aligned}y(n)&=b_0x(n)+b_1x(n-1)+\cdots+b_{M-1}x(n-M+1)\\&=b_0[x(n)+x(n-M+1)]+b_1[x(n-1)+x(n-M+2)]+\cdots\end{aligned} \tag{4-61}$$

图4－24表示了 M 为奇数和偶数两种情况下实现上述差分方程的方框图。

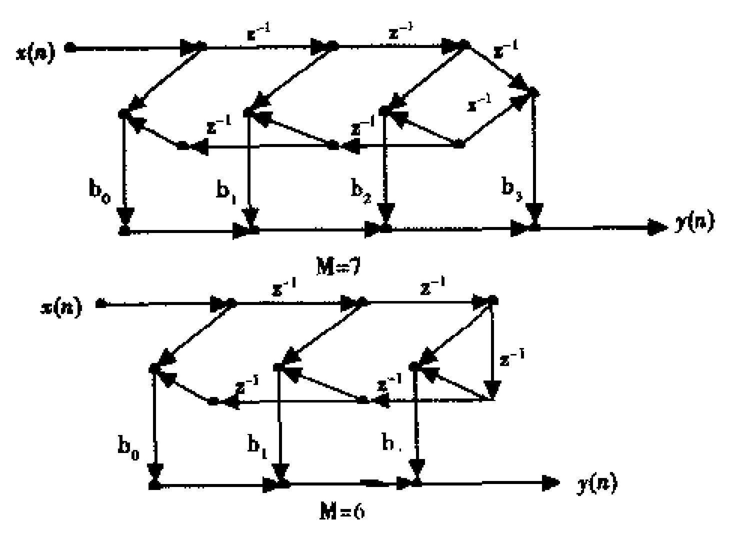


图4－24 FIR滤波器的线性相位型结构(对称冲激响应)

线性相位本质上仍然是直接型,它只是缩减了乘法计算量。因此,从MATLAB实现的方面考虑,线性相位结构等同于直接型。

4.3.2 数字滤波器的设计

1. IIR滤波器的设计

MATLAB信号处理工具箱中提供了丰富而简便的设计和实现FIR滤波器与IIR滤波器的方法,使原来繁琐的程序设计简化成为函数的调用,特别是滤波器的表达方式和滤波器形式之间相互转换显得十分简单方便。MATLAB为滤波器的设计和实现开辟了一片广阔的天地。

由于FIR滤波器和IIR滤波器不论是在理论上还是在设计方法上都有很大的不同,因此这里将分别讨论。

1)IIR滤波器设计简介

目前,IIR滤波器设计通常采用的方法是,利用已经很成熟的模拟滤波器的设计方法来进行设计。其具体的设计步骤如下:

(1)按照一定规则将给出的数字滤波器的技术指标转换为模拟低通滤波器的技术指标。

(2)根据转换后的技术指标设计模拟低通滤波器 $G(s)$。

(3)再按一定规则将 $G(s)$ 转换成 $H(z)$。

注意:如果设计的滤波器是高通、带通或带阻数字滤波器,则首先需要把它们的技术指标转换成低通模拟滤波器的技术指标,然后再按新的技术指标设计低通模拟滤波器 $G(s)$,最后再将 $G(s)$ 转换为 $H(z)$。

MATLAB工具箱提供了几种模拟滤波器的原型产生函数:Bessel低通模拟滤波器原型,Butterworth滤波器原型,Chebyshev(Ⅰ型、Ⅱ型)滤波器原型,椭圆滤波器圆形等不同的模拟滤波器原型。同时,它还提供了模拟低通滤波器向高通、带通和带阻的转变函数,模拟滤波器向数字滤波器转换的双线性变换法和冲激响应不变法,模拟IIR滤波器的阶数选择函数以及数字滤波器直接设计函数等等,为在MATLAB中设计IIR滤波器带来了极大的方便。

2)低通滤波器原型

在模拟滤波器设计中,首先给定滤波器的技术指标 α_1、α_2、Ω_1、Ω_2。其中,α_1 为通带达到的最大衰减,α_2 为阻带应达到的最小衰减,Ω_1 为通带截止角频率,Ω_2 为阻带截止角频率,假设需要设计的低通滤波器的 $G(s)$ 为:

$$G(s)=\frac{d_0+d_1s+\cdots+d_Ns^N}{c_0+c_1s+\cdots+c_Ns^N} \tag{4-62}$$

设计的目的是使 $G(s)$ 的对数幅频响应 $10\lg|G(j\Omega|^2$ 在 Ω_1 与 Ω_2 处分别达到 α_1、α_2 的要求。α_1、α_2 都是 Ω 的函数,它们的大小取决于 $|G(j\Omega|^2$ 的形状,为此定义了一个衰减

函数 $\alpha(\Omega)$，即：

$$\alpha(\Omega)=10\lg\left|\frac{X(j\Omega)}{Y(j\Omega)}\right|=10\lg\frac{1}{|G(j\Omega)|^2} \tag{4-63}$$

或者

$$|G(j\Omega)|^2=10^{-\alpha(\Omega)/10} \tag{4-64}$$

幅平方特性函数 $|G(j\Omega)|^2$ 在模拟滤波器设计中起着十分重要的作用，不同的 $|G(j\Omega)|^2$ 代表着不同的模拟滤波器原型，在 MATLAB 中也相应地提供了不同的原型设计函数。

◆ **Butterworth 滤波器**

Butterworth 滤波器具有通带内最大平坦的幅度特性，而且随着频率的升高呈单调减少。因此，Butterworth 滤波器又称为"最平"的幅频响应滤波器，而且 Butterworth 也是最简单的滤波器。

MATLAB 信号处理工具箱为低通模拟 Butterworth 滤波器的产生提供了函数 buttap()，其调用格式是：

```
[Z P K]=buttap(n);
```

调用后将返回一个 n 阶 Butterworth 滤波器的零点、极点和增益。

◆ **Chebyshev Ⅰ型滤波器**

Chebyshev Ⅰ型滤波器的特点是阻带内达到最大平滑，MATLAB 提供了产生低通模拟 Chebyshev Ⅰ型滤波器的函数 cheb1ap()，其调用格式是：

```
[Z P K]=cheb1ap(n,Rp);
```

调用后将返回一个 n 阶 Chebyshev Ⅰ型滤波器的零点、极点和增益。其中，Rp 为通带内的最大衰减。

◆ **Chebyshev Ⅱ型滤波器**

Chebyshev Ⅱ型滤波器的特点是通带内达到最大平滑，MATLAB 提供了产生低通模拟 Chebyshev Ⅱ型滤波器的函数 cheb2ap()，其调用格式是：

```
[Z P K]=cheb2ap(n,Rs);
```

调用后将返回一个 n 阶 Chebyshev Ⅱ型滤波器的零点、极点和增益。其中，Rs 为阻带内的最大衰减。

◆ **椭圆滤波器**

在 MATLAB 中，椭圆滤波器的实现函数是 ellipap()，其调用格式是：

```
[Z P K]=ellipap(n,Rp,Rs);
```

调用该函数后将返回一个 n 阶椭圆低通模拟滤波器圆形的零点、极点和增益。其中，Rs 为阻带内的最小衰减，Rp 为该滤波器通带内的最大衰减。

◆ **Bessel 滤波器**

Bessel 滤波器的实现函数为 besselap()，其调用格式为：

```
[Z P K]=besselap(n);
```

该函数用于返回一个 n 阶 Bessel 滤波器的零点、极点和增益。

3)低通、高通、带通及带阻滤波器的设计

模拟低通、高通、带通及带阻数字滤波器设计的主要方法是，先将需要设计的数字滤

波器技术指标通过某种频率转换关系转换成模拟低通滤波器的技术指标，并依据这些技术指标设计出低通滤波器的转移函数，然后再依据频率转换关系变成所要设计的滤波器的转移函数。

MATLAB 信号处理工具箱提供了表 4－5 所示四种转换函数。表 4－5 中同时还给出了低通、高通、带通及带阻滤波器的变换与设计公式。

表 4－5 低通、高通、带通及带阻滤波器的变换与设计公式

类型	变换函数公式	相应的设计公式
低通	$z^{-1}=\frac{p^{-1}-\alpha}{1-\alpha p^{-1}}$	$\alpha=\frac{\sin(\frac{\theta_c-\omega_c}{2})}{\sin(\frac{\theta_c+\omega_c}{2})}$ ω_c 为要求的截止频率
高通	$z^{-1}=\frac{p^{-1}+\alpha}{1+\alpha p^{-1}}$	$\alpha=\frac{\cos(\frac{\theta_c-\omega_c}{2})}{\cos(\frac{\theta_c+\omega_c}{2})}$ ω_c 为要求的截止频率
带通	$z^{-1}=-\frac{p^{-2}-\frac{2\alpha k}{k+1}p^{-1}+\frac{k-1}{k-1}}{\frac{k-1}{k+1}p^{-2}-\frac{2\alpha k}{k+1}p^{-1}+1}$	$\alpha=\frac{\cos[(\omega_{c1}+\omega_{c2})/2]}{\cos[(\omega_{c2}-\omega_{c1})/2]}$ $k=c\tan[(\omega_{c2}-\omega_{c1})/2]\tan(\theta_c)/2$ ω_{c1}、ω_{c2}是要求的上、下截止频率
带阻	$z^{-1}=-\frac{p^{-2}-\frac{2\alpha k}{k+1}p^{-1}+\frac{1-k}{k+1}}{\frac{1-k}{k+1}p^{-2}-\frac{2\alpha k}{k+1}p^{-1}+1}$	$\alpha=\frac{\cos[(\omega_{c1}+\omega_{c2})/2]}{\cos[(\omega_{c2}-\omega_{c1})/2]}$ $k=\tan[(\omega_{c2}\ \ \omega_{c1})/2]\tan(\theta_c)/2$ ω_{c1}、ω_{c2}是要求的上、下截止频率

◆ **从低通向低通的转换**

以下两个函数都是用于把模拟低通滤波器转换成频率为 ω_n 的低通滤波器，只是表达形式不同而已。

```
[AT BT CT DT]=lp2lp(A B C D Wn)
[numt dent]=lp2lp(num den Wn)
```

◆ **从低通到高通的转换**

以下两个函数都是把原来的低通滤波器原型转换成截止频率为 ω_0 的高通滤波器。

```
[AT BT CT DT]=lp2hp(A B C D W0)
[numt dent]=lp2hp(num den W0)
```

◆ **从低通到带通的转换**

以下两个函数把低通滤波器原型转换成截止频率为 ω_0、带宽为 B_ω 的带通滤波器。

```
[AT BT CT DT]=lp2bp(A B C D W0 Bw)
[numt dent]=lp2bp(num den W0)
```

◆ **从低通到带阻的转换**

以下两个函数把低通滤波器原型转换成中心频率为 ω_0、带宽为 B_ω 的带通滤波器。

[AT BT CT DT] = lp2bs(A B C D W0 Bw)

[numt dent] = lp2bs(num den W0)

4)IIR 数字滤波器阶数的选择

滤波器阶数的选择在整个滤波器设计中有着十分重要的地位和作用。在设计滤波器时，一般先进行阶数选择，再用返回的阶数和固有频率进行滤波器的设计。下面介绍几种滤波器阶数选择函数。

◆ **Butterworth 滤波器阶数选择函数**

下面两个 buttord 函数用来选择 Butterworth 滤波器的阶数：

[N Wn] = buttord(Wp Ws Rp Rs)

该函数用于返回符合要求性质的滤波器最小阶数 N 以及 Butterworth 滤波器固有频率 ω_n(即 3dB)，设计的要求是在通带内的衰减不超过 Rp，在阻带内的衰减不小于 Rs，通带和阻带的截止频率分别为 ω_p 和 ω_s，它们是归一化的频率，范围是[0 1]，对应 π 弧度。

[N Wn] = buttord(Wp Ws Rp Rs 's')

该函数用于返回符合要求性质的滤波器最小阶数 N，以及 Butterworth 滤波器固有频率 ω_n。其中，ω_p 和 ω_s 以弧度为单位，如果 Rp = 3dB，则固有频率 ω_n 等于通带截止频率 ω_p。

◆ **Chebyshev Ⅰ型滤波器阶数选择函数**

以下的 cheb1ord 函数用于选择 Chebyshev Ⅰ型滤波器的阶数：

[N Wn] = cheb1ord(Wp Ws Rp Rs)

该函数用于返回符合要求性质的滤波器最小阶数 N，以及 Chebyshev Ⅰ型滤波器固有频率 ω_n(即 3dB)，设计的要求是在通带内的衰减不超过 Rp，在阻带内的衰减不小于 Rs，通带和阻带的截止频率分别为 ω_p 和 ω_s，它们是归一化的频率，范围是[0, 1]，对应 π 弧度。

[N Wn] = cheb1ord(Wp Ws Rp Rs 's')

该函数用于返回符合要求性质的滤波器最小阶数 N，以及 ChebyshevⅠ型滤波器固有频率 ω_n。其中，ω_p 和 ω_s 以弧度为单位，如果 $Rp = 3dB$，则固有频率 ω_n 等于通带截止频率 ω_p。

◆ **Chebyshev Ⅱ型滤波器阶数选择函数**

以下的 cheb2ord 函数用于选择 Chebyshev Ⅱ型滤波器的阶数：

[N Wn] = cheb2ord(Wp Ws Rp Rs)

该函数用于返回符合要求性质的滤波器最小阶数 N，以及 ChebyshevⅡ型滤波器固有频率 ω_n(即 3dB)，设计的要求是在通带内的衰减不超过 Rp，在阻带内的衰减不小于 Rs，通带和阻带的截止频率分别为 ω_p 和 ω_s，它们是归一化的频率，范围是[0, 1]，对应 π 弧度。

[N Wn] = cheb2ord(Wp Ws Rp Rs 's')

该函数用于返回符合要求性质的滤波器最小阶数 N，以及 ChebyshevⅡ型滤波器固有频率 ω_n。其中，ω_p 和 ω_s 以弧度为单位，如果 Rp = 3dB，则固有频率 ω_n 等于通带截止频率 ω_p。

◆ **椭圆滤波器阶数选择函数**

以下的 ellipord 函数用于选择椭圆滤波器的阶数：

[N Wn] = ellipord(Wp Ws Rp Rs)

该函数用于返回符合要求性质的滤波器最小阶数 N，以及椭圆滤波器固有频率 ω_n(即 3dB)，设计的要求是在通带内的衰减不超过 Rp，在阻带内的衰减不小于 Rs，通带和阻

带的截止频率分别为 ω_p 和 ω_s，它们是归一化的频率，范围是[0, 1]，对应 π 弧度。

[N Wn] = ellipord(Wp Ws Rp Rs 's')

该函数用于返回符合要求性质的滤波器最小阶数 N，以及椭圆滤波器固有频率 ω_n。其中，ω_p 和 ω_s 以弧度为单位，如果 Rp = 3dB，则固有频率 ω_n 等于通带截止频率 ω_p。

5）冲激响应不变法

冲激响应不变法的基本原理是从滤波器的冲激响应出发，对具有传递函数 $G(s)$ 的模拟滤波器的冲激响应 $g(t)$，以周期 T 采样，得到离散序列 $g(nT)$ 作为数字滤波器的冲激响应。冲激响应不变法具有以下特点：

- 模拟频率与数字频率之间的转换关系是线性的，并保持了模拟滤波器的时域特性，这是冲激响应不变法的优点。
- 当模拟滤波器的频率响应不是严格限带时，则用冲激响应不变法设计出的数字滤波器在频域将出现混叠现象，这是冲激响应不变法的缺点。同时，使得它的应用受到了限制，即当 $G(j\Omega)$ 不严格限带或在时域 $g(t)$ 变化不太平稳、而设计的性能要求又较多时，不宜使用这种方法。

MATLAB 为冲激响应不变法提供了 impinvar()函数，这个函数用于实现从模拟到数字的转换。函数的调用格式是：

[Bz Az] = impinvar(B A Fs)

该函数用于把具有[B A]模拟滤波器的传递函数模型转换成采样频率为 Fs 的数字滤波器的传递函数模型[Bz Az]。如果没有确定采样频率 Fs 时，函数默认为 1Hz。

6）双线性变换法

为了克服冲激响应不变法产生的频率混叠现象，需要使 S 平面与 Z 平面建立一一对应的单值关系，即求出 $s = f(z)$，然后将它代入 $G(s)$，即可以得到 $H(z)$。

双线性变换法的特点是：

- 模拟滤波器的传递函数 $G(s)$ 经双线性变换后，不存在幅度频率特性混叠失真现象，因而对 $G(s)$ 要求放宽，故适用范围广，且设计简单，容易实现。
- 模拟滤波器通过双线性变换后，会出现相位频率特性失真现象，所以当对滤波器的相位特性有较为严格的要求时，不宜采用这种设计方法。

MATLAB 为实现双线性变换提供了 bilinear()函数，其调用格式有三种，分别为：

格式一：

[Zd Pd Kd] = bilinear(Z, P, K, Fs)

该函数格式用于把模拟滤波器零点、极点模型转换成数字滤波器的零点、极点模型，其中的 Fs 是采样频率。

格式二：

[NUMd DENd] = bilinear(NUM, DEN, Fs)

该函数格式用于把模拟滤波器的传递函数模型转换成数字滤波器的传递函数模型，其中的 Fs 是采样频率。

格式三：

[Ad Bd Cd Dd] = bilinear(A, B, C, D, Fs)

该函数格式用于把模拟滤波器的状态方程模型转换成数字滤波器的状态方程模型，

其中的 Fs 是采用频率。

以上三种滤波器还可以另外限定预畸变频率 Fp,即在进行双线性变换前,对采样频率进行畸变,以保证频率冲激响应在双线性变换前后具有良好的单值映射关系,具体的预畸变过程如下:

```
Fp = 2 * PI * Fp
Fs = Fp/tan(Fp/Fs/2)
```

7)IIR 数字滤波器设计

在 MATLAB 中,IIR 数字滤波器的设计过程如下:

(1)按一定规则将给出的数字滤波器的技术指标转换成模拟低通滤波器的技术指标。

(2)根据转换后的技术指标,使用滤波器阶数选择函数,确定滤波器最小阶数 N 以及固有频率 ω_n。

(3)运用滤波器最小阶数 N 产生模拟低通滤波器原型。

(4)运用滤波器固有频率 ω_n 把模拟低通滤波器原型转换成模拟低通、高通、带通或带阻滤波器。

(5)运用冲激响应不变法或是双线性变换法把模拟滤波器转换成数字滤波器。

上述的设计过程是典型滤波器的设计过程,另外,MATLAB 还提供了多种直接设计 IIR 滤波器的函数和方法。下面将这些函数和方法总结如下:

◆ **典型滤波器的设计**

典型滤波器的设计函数有如下几类:

- 模拟低通滤波器原型的创建函数。包括 buttap、cheb1ap、cheb2ap、ellipap 和 besselap 等。
- 频率转换函数。包括 lp2lp、lp2hp、lp2bp 和 lp2bs 等。
- 离散化处理函数。包括 impinvar 和 bilinear 等。

◆ **完全滤波器设计**

完全滤波器设计函数有五种,即 butter、Cheby Ⅰ、Cheby Ⅱ、ellip 和 besself 等。

◆ **直接设计方法**

直接设计滤波器的方法有:Yule_Walk 法、Prony 法、线性预测法、反向频率设计法和 Steiglitz_Mcbride 法等。

2. FIR 滤波器的设计

由于 FIR 滤波器具有 IIR 滤波器所没有的线性相位特性,这使得信号在传输时不会发生明显的相位失真。所以,FIR 滤波器在工程中得到了广泛的应用,设计 FIR 滤波器常用的方法是窗口法和频率采样法。

1)窗口法

基于窗函数的 FIR 数字滤波器的设计步骤为:

(1)设 $H_d(e^{j\omega})$是所要求的 FIR 滤波器理想频率响应有:

$$H_d(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h_d(n)e^{-j\omega n} \tag{4-65}$$

它与单位脉冲相应序列 $h_d(n)$是一对傅立叶变换对，即：

$$h_d(n)=\frac{1}{2\pi}\int_{-\pi}^{\pi}H_d(e^{j\omega})e^{j\omega n}\mathrm{d}\omega \tag{4-66}$$

提示：一般设理想低通滤波器的通带截止频率为 ω_p，阻带截止频率为 ω_s，其幅频特性满足以下条件：

$$|H_d(e^{j\omega})|=\begin{cases}1 & 0\leqslant|\omega|\leqslant\omega_p\\ 0 & \omega^s\leqslant|\omega|\leqslant\pi\end{cases} \tag{4-67}$$

(2)由性能指标确定窗函数 $W(n)$和窗口长度 N，由过渡带宽近似于窗函数主瓣宽求得窗口长度 N；

(3)由 $h(n)=h_d(n)W(n)$求得实际滤波器的单位脉冲响应 $h(n)$，$h(n)$为所设计的 FIR 滤波器系数向量；

(4)检验滤波器的性能。

采用窗口法进行 FIR 滤波器设计的 MATLAB 函数有以下两种：

◆ **标准型 FIR 滤波器设计函数**

函数 fir1 采用经典窗函数设计线性相位 FIR 数字滤波器，它可以用来设计标准的低通、高通、带通和带阻等类型的 FIR 滤波器，函数的调用格式如下：

```
b=fir1(n Wn);
b=fir1(n Wn 'ftype');
b=fir1(n Wn window);
b=fir1(n Wn 'ftype' window);
```

其中，n 为 FIR 滤波器的阶数，对于高通、带阻滤波器，n 应该取偶数；Wn 为滤波器的截止频率，0<Wn<1，对于带通、带阻滤波器，Wn=[W1 W2]，且 W1<W2，对于多带滤波器，Wn=[W1 W2 W3…]，且满足 0<W1<W2<…<π；b 是 FIR 滤波器的系数向量，长度为 n+1；'ftype'是滤波器的类型，默认时为低通或带通滤波器，'high'为高通滤波器，'stop'为带阻滤波器。window 为窗函数，是长度为 n+1 的列向量。MATLAB 提供的窗函数有：boxcar(矩形窗函数)，hamming(海明窗函数)，barlett(巴特洛特窗函数)，hanning(汉宁窗函数)，triang(三角窗函数)，kaiser(凯塞窗函数)，blackman(勃莱克曼窗函数)以及 chebwin(切比雪夫窗函数)。

用函数 fir1 设计的 FIR 滤波器的群延时为 n/2。

◆ **多带 FIR 滤波器设计函数**

多带 FIR 滤波器设计函数 fir2 用于设计具有任意形状频率响应的 FIR 数字滤波器，函数的调用格式为：

```
b=fir2(n, f, m);
b=fir2(n, f, m, window);
b=fir2(n, f, m, npt);
b=fir2(n, f, m, npt, window);
b=fir2(n, f, m, npt, lap);
b=fir2(n, f, m, npt, lap, window);
```

其中，n 为滤波器的阶数；f 和 m 分别为滤波器的期望幅频响应的频率分量和幅值分

量,取值均在0～1之间,并且它们的长度相同;window为窗函数,是长度为n+1的列向量,默认时为Hamming窗;npt为对频率响应进行内插的点数,默认时为512;lap定义一个区域尺寸,函数fir2在重复频率点周围建立这个区域并提供光滑、陡峭的过渡频率响应,默认值为25;b为FIR滤波器的系数向量。

2) FIR滤波器的优化设计

函数firs是函数fir1和函数fir2的扩展,其设计准则是利用最小二乘法使期望的频率响应和实际频率响应之间的整体误差最小。函数remez采用Parks-McClellan算法进行FIR滤波器的设计。函数remez是最流行和应用最广泛的一种FIR滤波器设计方法,两函数的调用格式是相同的,其格式是:

```
b=firs(n, f, a);
b=firs(n, f, a, w);
b=firs(n, f, a, 'd');
b=firs(n, f, a, w, 'd');
b=firs(n, f, a, w, 'h');
b=remez(n, f, a);
b=remez(n, f, a, w);
b=remez(n, f, a, 'd');
b=remez(n, f, a, w, 'd');
b=remez(n, f, a, w, 'h');
```

其中,n为滤波器的阶数;f为滤波器的期望幅频响应的频率分量,取值在0～1之间,且必须是递增向量;a为滤波器的期望幅频响应的幅值分量,a和f的长度必须相同,且为偶数;w为各频带的权向量,其长度应为f和a长度的一半,且一个频带必须对应一个权值;'d'为选择项,表示所设计的FIR滤波器具有微分器的作用;当添加'h'或'Hdbert'选项时,表示用函数firs和remez设计奇对称FIR滤波器,这种滤波器具有希尔伯特变换器的特性;b为FIR滤波器的系数向量,长度为n+1。

3) 约束最小二乘FIR滤波器设计

约束最小二乘(CLS)FIR滤波器设计函数因对幅频响应的过渡带没有明确定义而无需指明过渡带的位置,只需给出频率响应的截止频率(高通、低通、带通或带阻)或通带与阻带的边界(多带的情况),它的关键特征是在给定的滤波器幅频响应最大允许波纹的上下阈值约束条件下,使实际滤波器的幅频响应在整个频率范围内平方误差最小。

约束最小二乘FIR滤波器设计的函数及其调用格式为:

```
b=fircls(n, f, a, wp, t0);
b=fircls(n, f, a, wp, t0, 'flap');
```

其中,n为滤波器的阶数;f为滤波器的期望幅频响应的频率分量,它是标准化频率,在0～1之间取值,且第一个值为0,最后一个值为1;a为滤波器的期望幅频响应的幅值分量,它的长度为length(f)－1;wp和t0分别为每一个频带的上边界频率和下边界频率,均为长度等于a的向量;'flap'为可选项,用于监视滤波器设计,其取值有三种:'trace'表示文字跟踪显示,'plot'表示绘制滤波器幅频图、群延时和零极点图,'both'表示上述两种方式同时采用;b为FIR滤波器的系数向量。

函数fircls1用于低通和高通线性相位滤波器的约束最小二乘滤波器的设计。其调用

格式为：

```
b=fircls1(n, w0, dp, ds);
b=fircls1(n, w0, dp, ds, wt);
b=fircls1(n, w0, dp, ds, wt, 'flag');
b=fircls1(n, w0, dp, ds, 'high');
b=fircls1(n, w0, dp, ds, wt, 'high');
b=fircls1(n, w0, dp, ds, wt, 'high', 'flag');
```

其中，前三个函数用于低通 FIR 滤波器的设计，后三个函数用于高通 FIR 滤波器的设计。n 为滤波器的阶数；w0 为滤波器的截止频率，取值在 0～1 之间；dp 为通带离幅值 1 的最大偏差；ds 为阻带离幅值 0 的最大偏差；'flag'为设计监测标志，其含义与在函数 fircls 中的含义相同；wt 为一个定义频率，用于确保设计的滤波器满足通带或阻带的边界要求，若 0＜wt＜w0＜1，则在 0＜w＜wt 范围内滤波器幅值允许误差在 dp 内，若 0＜w0＜wt＜1，则在 wt＜w＜1 范围内滤波器幅值允许误差在 ds 内；b 为 FIR 滤波器的系数向量。

除了上述三种方法以外，MATLAB 信号处理工具箱还提供了可以设计任意复响应和非线性相位的 FIR 滤波器的函数 cremez 和设计升余弦低通 FIR 滤波器的函数 firrcos。

4.4 信号处理的交互式工具——SPTool

SPTool(Signal Processing Tool)是 MATLAB 信号处理工具箱中的一个具有交互式图形界面的信号处理工具。该工具包含了信号处理工具箱中的大部分函数，可以方便快捷地完成常规的数字信号处理。由于采用的是图形界面，特别适合初学者学习。本节将对 SPTool 及其使用方法作简要介绍，以引导读者去了解、使用这一工具及进一步挖掘它的强大功能，为学习和研究数字信号处理打下良好的基础。

4.4.1 SPTool 简介

当在 MATLAB 的命令窗口中键入命令“sptool”并回车后，便会打开 SPTool 窗口，如图 4－25 所示。

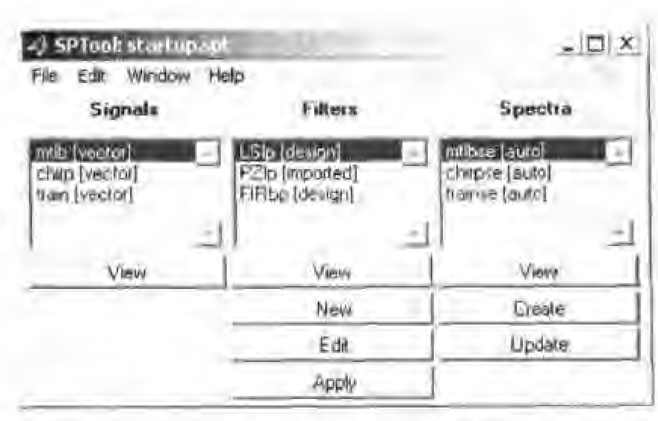


图 4－25 SPTool 用户界面

SPTool 是一个图形环境，可用于信号浏览、滤波器的设计和分析以及频谱分析。它主要由三个交互式信号处理模块构成：

- Signal Browser(信号浏览器)模块：主要用于各种数字信号的显示、分析及打印等。
- Filters(滤波器分析与设计)模块：主要用于 FIR 和 IIR 数字滤波器的幅频及相位响应、阶跃及冲激响应等的查看和各种滤波器的分析与设计。
- Spectra(功率谱分析)模块：主要用于各种数字信号的功率谱分析。

SPTool 图形窗口包括以下四个菜单：

- File(文件)菜单：该菜单包括 Open Session(会话文件的打开)、Import(信号、滤波器和功率谱的导入)、Export(信号、滤波器和功率谱的导出)、Save Session 和 Save Session As(会话文件的保存)以及 Preference(参数的选择)和 Close(关闭)等命令。
- Edit(编辑)菜单：该菜单包括 Duplicate(信号、滤波器和功率谱的复制)与 Clear(清除)、Name(对象的命名)和 Sampling Frequency(抽样频率的设置)等命令。
- Window(窗口)菜单：该菜单的主要功能是完成 SPTool 的各个窗口与 MATLAB 的命令窗口之间的切换，要激活某个窗口，只需在 Window 菜单下选择这个窗口的名称即可。
- Help(帮助)菜单：该菜单包括 Overview(SPTool 的概述)和 Context Sensitive(上下文相关帮助)等命令。

4.4.2 信号浏览器

SPTool 的 Signal Browser(信号浏览器)提供了一个交互式的信号处理环境，如图 4-26所示。它可以实现以下功能：

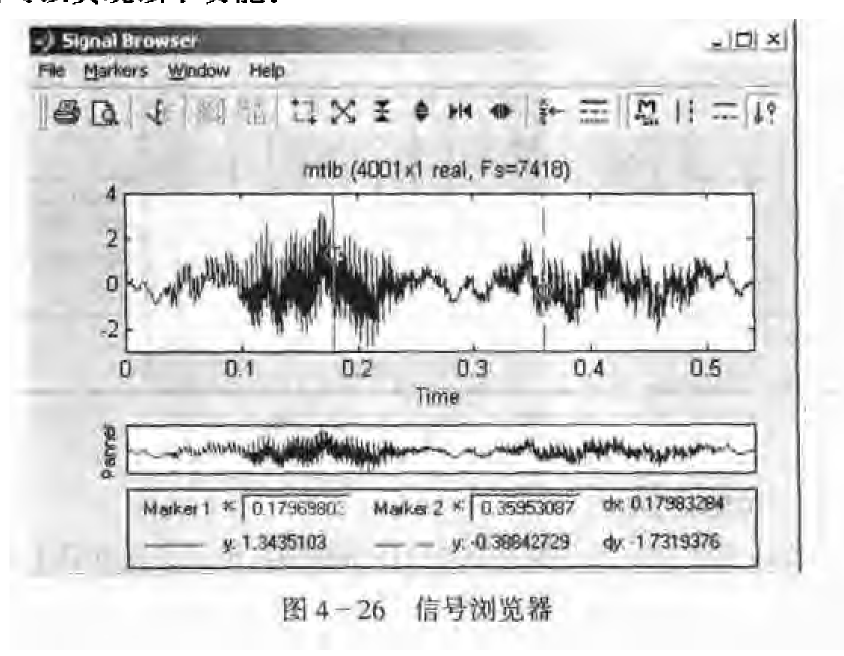


图 4-26　信号浏览器

- 查看和分析序列信号与数组信号。
- 放大信号某一局部范围，以便于查看信号的细微部分。

- 获取信号的特征向量。
- 打印信号。

在信号浏览器窗口中，中间是信号显示区域；信号显示区域上面是比例控制工具栏，包括 Mouse Zoom(鼠标指定区域的放大)、Full View(全部信号显示)、Zoom Out - Y(Y 轴放大)、Zoom In - Y(Y 轴缩小)、Zoom Out - X(X 轴放大)和 Zoom In - X(X 轴缩小)等按钮。另外，Rulers 为标尺和线条显示控制；Panner 为信号带，它始终显示信号的全景图。

当要浏览某个信号时，先在 SPTool 窗口中选择一个或多个信号，然后单击 Signal 面板下的 View 按钮，激活信号浏览器，如图 4 - 25 所示。这时，在信号显示区域中将用不同颜色显示出不同信号，并在此区域的上方显示出所有信号的名称、大小、类型和抽样频率。在 Panner 区域中将显示出信号的全景图。

4.4.3 滤波器分析与设计

在 SPTool 中有一个滤波器设计器——Filter Designer，如图 4 - 27 所示。它提供了交互式的滤波器设计环境，能根据滤波器的幅值和零极点图的设置以及滤波器设计方法的选择而自动地进行 FIR 和 IIR 数字滤波器的设计。在 Filter Designer 中，可设计任意长度的、多种类型的、低通、带通、带阻、高通以及多带 FIR 和 IIR 数字滤波器。

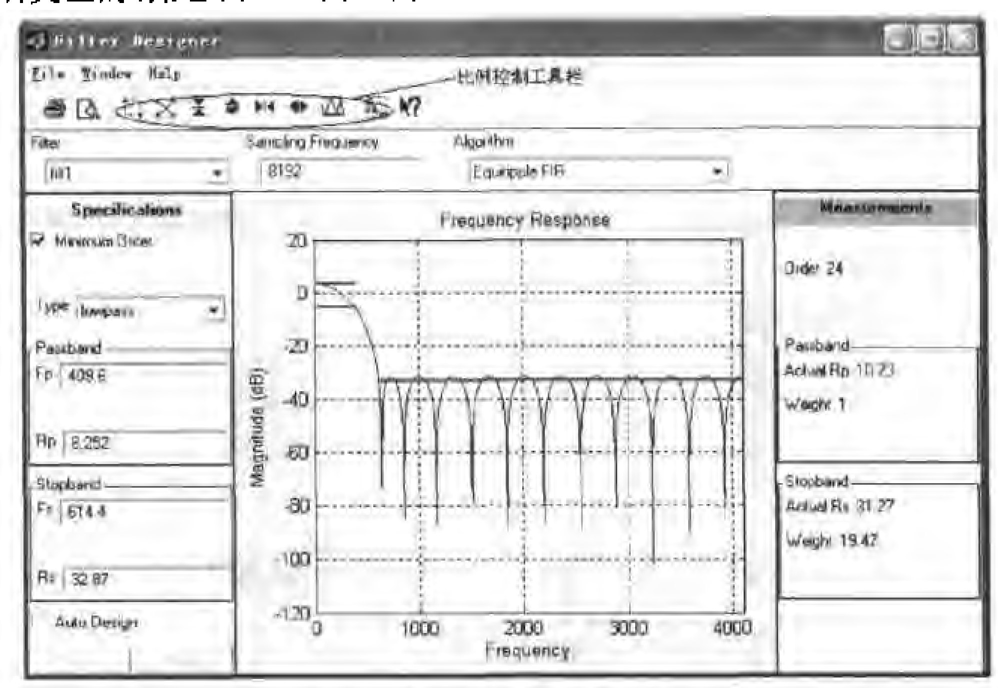


图 4 - 27 Filter Designer 窗口

Filter Designer 主要提供以下功能：

- 能够设计具有标准频率带宽结构的 IIR 滤波器，设计方法包括 Butterworth、Chebyshev I 和 Chebyshev II 等。
- 能够设计具有标准频率带宽结构的 FIR 滤波器，设计方法包括 equiripple()等波纹、least square(最小方差)和 Kaiser 窗等。
- 通过零极点编辑器，可以实现具有任意频率带宽结构的 FIR 和 IIR 滤波器的设计。

- 通过调整传递函数零点的图形位置，可以实现滤波器的再设计。

Filter Designer 窗口主要由以下几部分组成：

- 比例控制工具栏：用于查看滤波器幅频响应图和零极图的局部特征。
- Specifications 面板：用于查看和修改滤波器的参数或零极点的位置。
- Frequence Response(滤波器幅频响应)显示区域：它位于窗口的正中，包含滤波器通带和阻带的 Specification 线条，用于以图形方式调整幅频响应参数。
- Measurements 面板：用于查看滤波器幅频响应特征量以及滤波器的稳定性指标。
- Algorithm(滤波器设计方法选择)下拉列表框：用于选择 IIR 和 FIR 滤波器的设计方法。
- 当前滤波器的零极点显示图：当在 Algorithm 中选择 Pole/Zero Editor 时，在主显示区域中将会出现当前滤波器的零极点图。

当要设计滤波器时，只需将所需要设计的滤波器的参数输入到相应位置，并选择设计方法，就可设计出相应的滤波器，并显示出幅频响应曲线。

在 SPTool 中，除了有一个滤波器设计器，还有一个 Filter Viewer(滤波器浏览器)，如图 4－28 所示，它主要用于滤波器的分析。

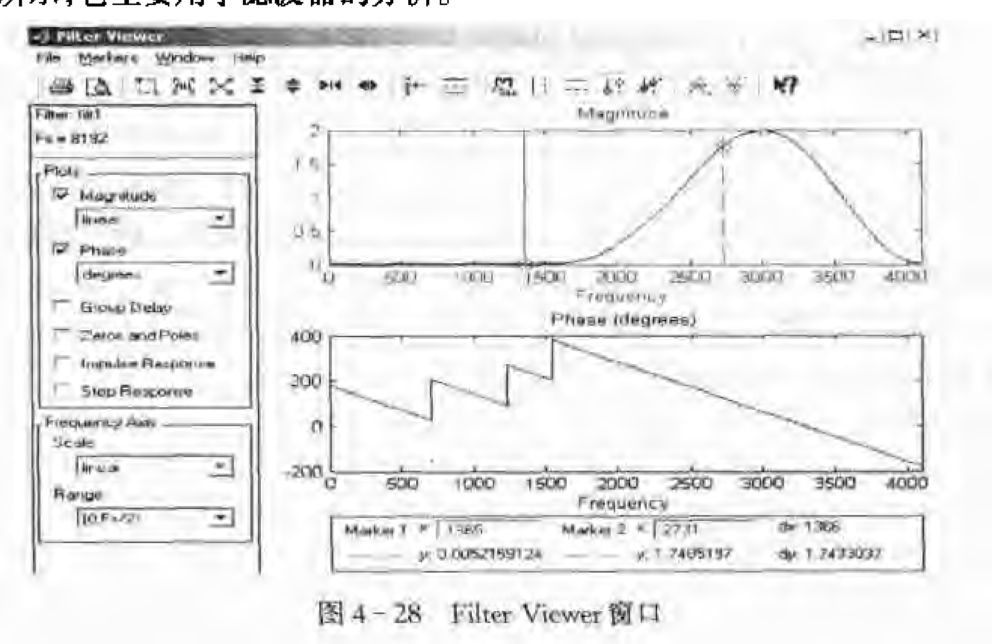


图 4－28　Filter Viewer 窗口

Filter Viewer 窗口主要由以下几部分组成：

- 比例控制工具栏：用于查看滤波器响应的细节特征，与图 4－27 所示的比例控制工具栏相同。
- Plots 面板：主要用于选择主显示区域中显示的子图。
- Frequency Axis 面板：用于设置主显示区域中显示的图形的 X 坐标尺度。
- Frequency 信息面板：用于显示选定滤波器的有关信息。
- Rules 和线条显示控制的面板：用于进行信号的标定和比较。
- 主显示区域：用于显示选定滤波器的一个或多个频域图形。

Filter Viewer 能完成以下功能：

- 浏览一个或多个滤波器的幅频响应图。

- 浏览一个或多个滤波器的相位响应图。
- 浏览一个或多个滤波器的零极点图。
- 浏览一个或多个滤波器的脉冲响应图。
- 浏览一个或多个滤波器的阶跃响应图。
- 图形的比例控制,可查看滤波器响应的细节特征。
- 改变选定的图形参数和显示特征。
- 实现滤波器响应的多个特征量的标定。

将滤波器设计器和浏览器结合使用,可以非常方便快捷地完成给定滤波器的设计。

4.5 信号处理仿真实例

在 4.3 节中讨论了 IIR 数字滤波器和 FIR 数字滤波器的设计方法及 MATLAB 信号处理工具箱提供的各种滤波器设计函数,本节将通过实例说明如何使用这些滤波器设计函数进行各种数字滤波器的设计。

例 4-8　设计一个 3 阶低通 IIR 模拟滤波器,要求滤波器在通带内的最大衰减为 3dB,阻带内的最小衰减为 40dB。

分析:这是一个模拟原型滤波器设计实例。设计低通 IIR 模拟滤波器的方法包括:Butterworth、Chebyshev I、Chebyshev II、椭圆滤波器和 Bessel 滤波器等,各种设计方法的函数调用格式不一样。经过比较,可以看出最适合本例的方法是采用椭圆滤波器设计法,因为它可同时指定通带和阻带的技术指标。为了更好地进行对比,本例采用 Butterworth、Chebyshev II 和椭圆滤波器设计法分别进行设计。

实现例 4-8 的 MATLAB 源代码如下:

```
% MATLAB program 4-8
% 3-order Analog filter design
clear all;
clc;
N=3;
Rp=3;
Rs=40;
[z1,p1,k1]=buttap(N);                % Butterworth filter
[z2,p2,k2]=cheb2ap(N,Rs);            % Chebyshev II filter
[z3,p3,k3]=ellipap(N,Rp,Rs);         % Elliptic filter,零极点形式转换成状态空间形式
[b1,a1]=zp2tf(z1,p1,k1);
[b2,a2]=zp2tf(z2,p2,k2);
[b3,a3]=zp2tf(z3,p3,k3);
figure;
[h1,w1]=freqs(b1,a1);
[h2,w2]=freqs(b2,a2);
[h3,w3]=freqs(b3,a3);
```

```
mag1 = 20 * log10(abs(h1));
plot(w1,mag1);
grid;                                  % 绘制频率响应图
figure;
mag2 = 20 * log10(abs(h2));
plot(w2,mag2);
grid;
figure;
mag3 = 20 * log10(abs(h3));
plot(w3,mag3);
grid;
```

程序运行结果如图 4－29(a)、(b)、(c)所示。

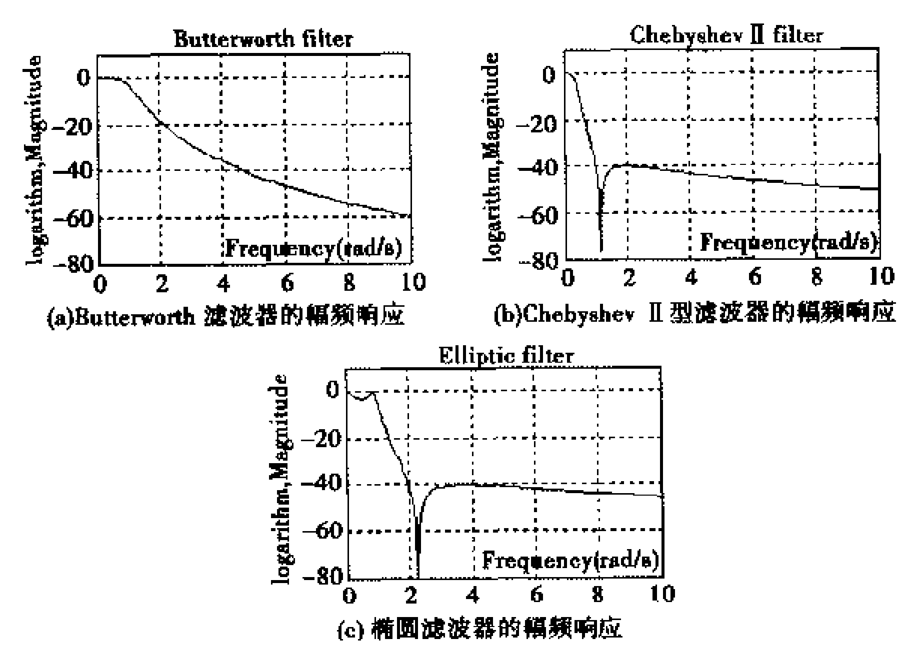


图 4－29　例 4－8 中滤波器的幅频响应

从采用三种不同方法设计出的滤波器的幅频响应曲线可以看出，这三种滤波器具有各自不同的特点：Butterworth 滤波器能在通带内达到最大限度的平坦，但在截止频率处的下降斜度较小；而 Chebyshev I 型滤波器和椭圆滤波器在截止频率处的下降斜度较大，其中，Chebyshev I 型滤波器在通带内为等波纹；椭圆滤波器在通带和阻带内均为等波纹。

例 4－9　设计一个 10 阶低通模拟滤波器，通带内最大衰减为 3dB，阻带内最小衰减为 60dB，截止频率为 6π 弧度，然后再把它转换成截止频率为 40π 弧度的高通滤波器，并绘制出它们的频率响应图。

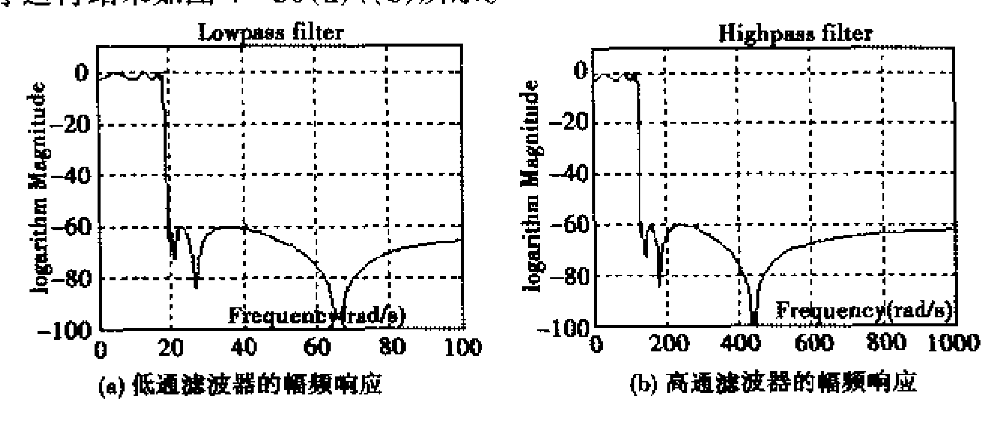
分析：本例说明了如何设计满足指定截止频率的低通滤波器和高通滤波器。由本章 4.3 节可知，MATLAB 信号处理工具箱中没有给出截止频率的低通和高通滤波器函数，所以只有先设计一低通原型滤波器，再将它转换成所需的截止频率的低通和高通滤波器。由于本例中还有通带、阻带性能指标的要求，因此，选用椭圆滤波器函数进行低通原型滤波器设计，然后再把得到的低通原型滤波器通过低通到低通、低通到高通滤波器的转换函

数转换成所需的滤波器。

实现例 4－9 的 MATLAB 源代码如下：

```
%MATLAB program 4-9
% 10-order Analog filter design
clear all;
clc;
N=10; Rp=3; Rs=60;
[z,p,k]=ellipap(N,Rp,Rs);                    % 设计模拟原型低通滤波器
[A,B,C,D]=zp2ss(z,p,k);
[At,Bt,Ct,Dt]=lp2lp(A,B,C,D,6*pi);           % 转换成所需的低通滤波器
[b,a]=ss2tf(At,Bt,Ct,Dt);
figure;
[h,w]=freqs(b,a);
mag = 20*log10(abs(h));
plot(w,mag);
grid;
[At1,Bt1,Ct1,Dt1]=lp2lp(A,B,C,D,40*pi);     % 模拟原型低通滤波器
[b1,a1]=ss2tf(At1,Bt1,Ct1,Dt1);              % 转换成所需的高通滤波器
figure;
[h1,w1]=freqs(b1,a1);
mag1 = 20*log10(abs(h1));
plot(w1,mag1);
grid;
```

程序运行结果如图 4－30(a)、(b)所示。



(a) 低通滤波器的幅频响应　　(b) 高通滤波器的幅频响应

图 4－30　例 4－9 中滤波器的幅频响应

说明：本例只给出了低通和高通滤波器的设计方法，如若设计带通或带阻，或者多带滤波器，可仿照本例的方法进行。

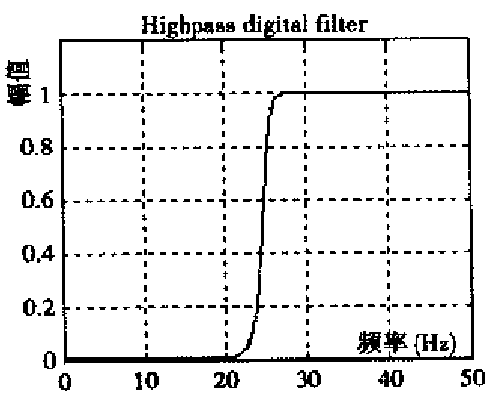
例 4－10　设计一个数字信号处理系统，抽样频率为 Fs＝500Hz。同时，在该系统中需要设计一个 Butterworth 型滤波器，通带内允许的最大衰减为 0.5dB，阻带内允许的最小衰减为 40dB，通带截止频率为 30Hz，阻带截止频率为 40Hz。

分析：该例说明了如何利用滤波器阶数选择函数进行所需的滤波器阶数选择。同时，也介绍了如何使用变换法将模拟滤波器变换成数字滤波器。

实现例4－10的MATLAB源代码如下：

```
%MATLAB program 4-10
% Digital filter design
clear all;clc;
Wp=2*pi*30;Ws=2*pi*40;                % 把数字滤波器的频率特征转换成模拟滤波器
                                      % 的频率特征
Rp=0.5; Rs=40; Fs=500;
[N,Wn]=buttord(Wp,Ws,Rp,Rs,'s');      % 选择滤波器的阶数
[z,p,k]=buttap(N);                    % 创建Butterworth低通滤波器原型
[A,B,C,D]=zp2ss(z,p,k);               % 从零极点形式转换成状态方程形式
[At,Bt,Ct,Dt]=lp2hp(A,B,C,D,Wn);      % 实现低通到高通滤波器类型的转换
[b,a]=ss2tf(At,Bt,Ct,Dt);             % 采用双线性变换法实现从模拟低通到数字高
                                      % 通滤波器的转换
[b1,a1]=bilinear(b,a,Fs);
figure;                               % 绘出频率响应曲线
[h,w]=freqz(b1,a1);
plot(w*Fs/(2*pi),abs(h));
grid;
xlabel('频率(Hz)');ylabel('幅值');title('Highpass digital filter');
figure;
plot(w*Fs/(2*pi),20*log10(abs(h)));
grid;
xlabel('频率(Hz)');ylabel('幅值(dB)');title('Highpass digital filter');
```

程序运行结果如图4－31所示。



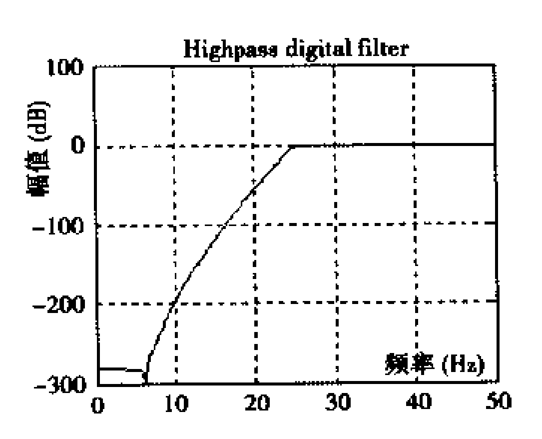


图4－31　例4－10中滤波器的幅频响应

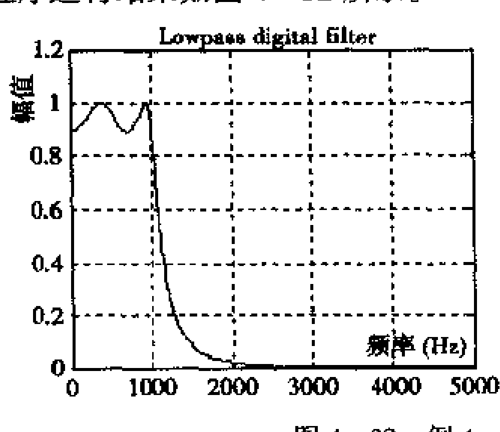
例4－11　用冲激响应不变法设计数字Chebyshev低通滤波器，给定的采样频率为Fs=10kHz，要求在频率小于1kHz的通带内，幅度特性下降小于1dB，在频率大于1.5kHz的阻带内衰减小于15dB。

分析：本例中对滤波器的设计要求在实际应用中常常会遇到，它给出的是数字滤波器的性能指标，其设计过程是一典型的IIR数字滤波器设计过程，因而需要按照IIR数字滤波器的设计步骤逐步进行。

实现例4－11的MATLAB源代码如下：

```
%MATLAB program 4-11
% Typical digital filter design
clear all;
clc;
Fs=1.0e+4;Fc=1.0e+3; Fst=1.5e+3;Rp=1; Rs=15;       % 把数字滤波器的频率特征转换
                                                   % 成模拟滤波器的频率特征
Wp=2*pi*Fc;
Ws=2*pi* Fst ;
[N,Wn]=cheb1ord(Wp,Ws,Rp,Rs,'s');                 % 选择滤波器的阶数
[z,p,k]=cheb1ap(N);                                % 创建 Chebyshev I 低通滤波器
                                                   % 原型
[A,B,C,D]=zp2ss(z,p,k);                            % 从零极点形式转换成状态方程
                                                   % 形式
[At,Bt,Ct,Dt]=lp2lp(A,B,C,D,Wn);                   % 实现低通到高通滤波器类型的
                                                   % 转换
[b,a]=ss2tf(At,Bt,Ct,Dt);                          % 采用双线性变换法实现从模拟
                                                   % 低通到数字高通滤波器的转换
[b1,a1]=impinvar(b,a,Fs);
figure;                                            % 绘出频率响应曲线
[h,w]=freqz(b1,a1);
plot(w*Fs/(2*pi),abs(h));
grid;
xlabel('频率(Hz)');
ylabel('幅值');
title('Highpass digital filter');
figure;
plot(w*Fs/(2*pi),20*log10(abs(h)));
grid;
xlabel('频率(Hz)');
ylabel('幅值(dB)');
title('Highpass digital filter');
```

程序运行结果如图 4－32 所示。



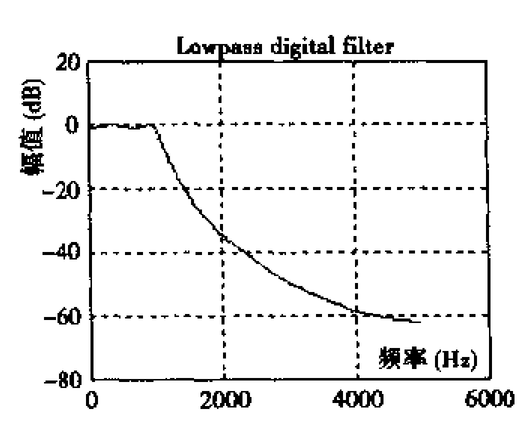


图 4－32　例 4－11 中滤波器的幅频响应

4.6　数字信号处理仿真模块

在 MATLAB 命令窗口中键入 simulink 命令，回车后便可进入 MATLAB 的动态仿真环境，并打开仿真模块库浏览器，如图 4－33 所示。通过单击 DSP Blockset 可进入 DSP Blockset（数字信号处理仿真子模块库），在 MATLAB 命令窗口中键入 dsplib 命令，同样可进入数字信号处理仿真子模块库，如图 4　34 所示。该子模块库包含 10 个子模块集，下面对其中的一些主要子模块集进行介绍。

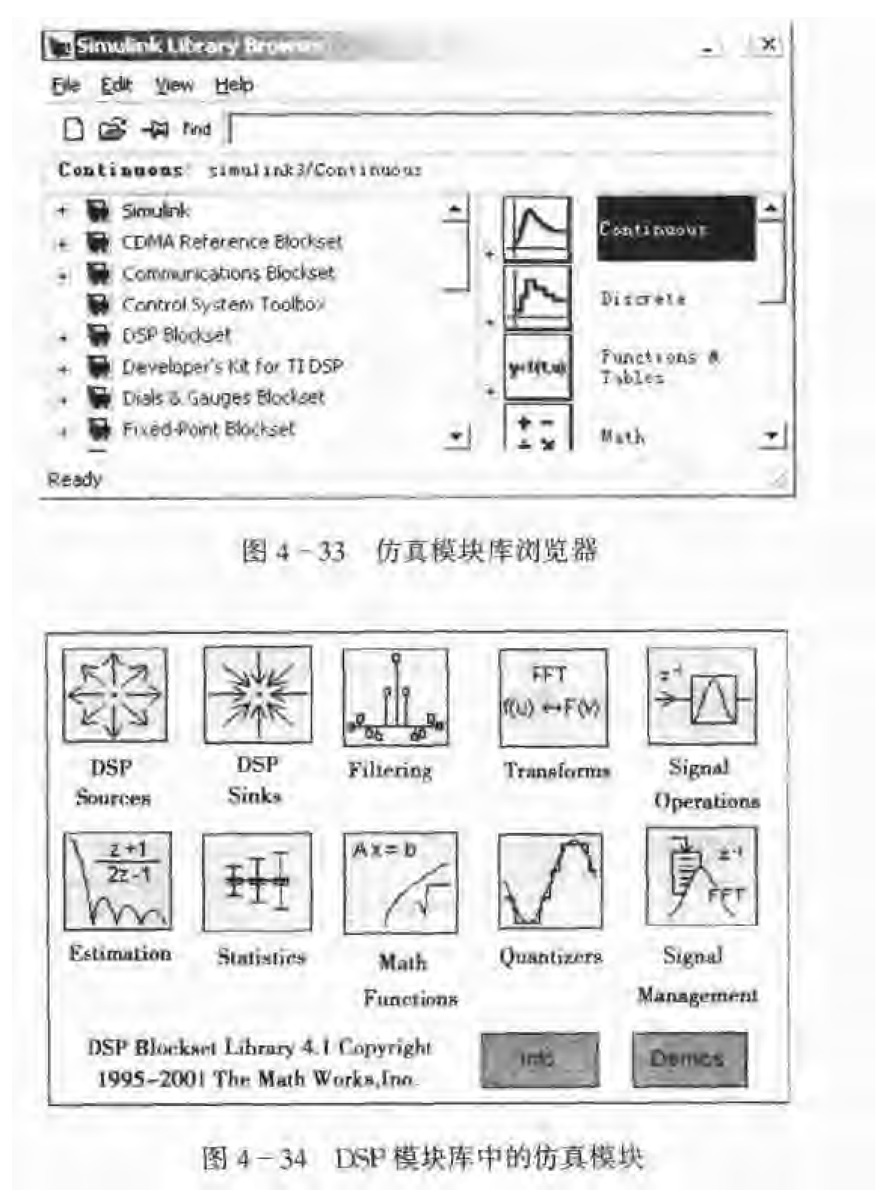


图 4－33　仿真模块库浏览器

图 4－34　DSP 模块库中的仿真模块

◆　**DSP Sources（DSP 信号源）子模块集**

如图 4－35 所示，DSP Sources（DSP 信号源）子模块集包含各种进行数字信号处理时需要的输入信号源。

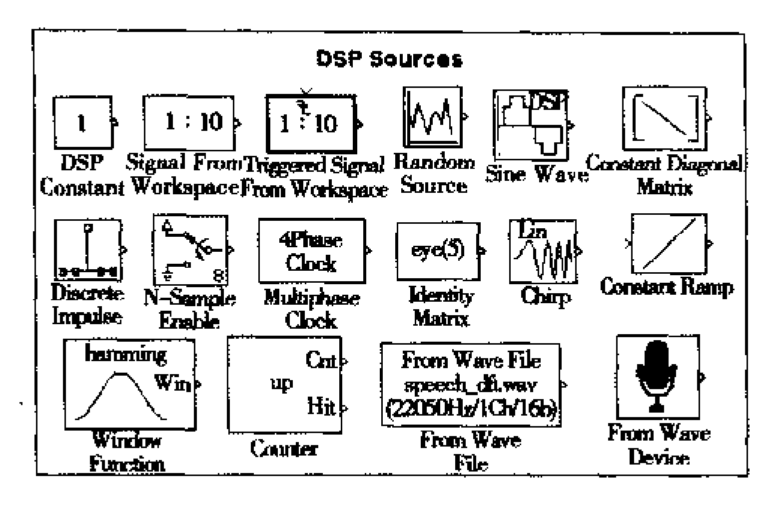


图 4 - 35　DSP 信号源子模块集

◆　**DSP Sinks(DSP 信号接收)子模块集**

如图 4 - 36 所示，DSP Sinks(DSP 信号接收)子模块集包含各种处理后的数字信号的接收模块。

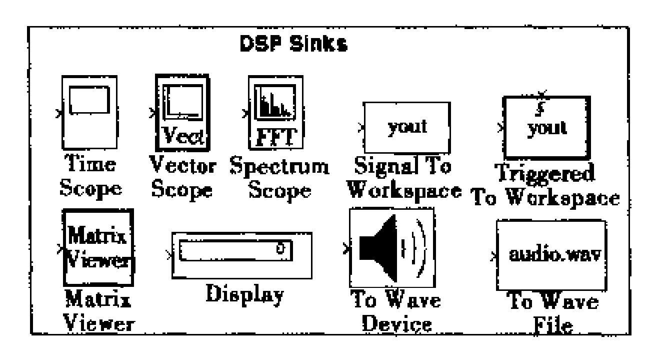


图 4 - 36　DSP 信号接收子模块集

◆　**Filtering(滤波器)子模块集**

在 Filtering(滤波器)子模块集中又包含了 Filter Designs(滤波器设计)子模块集(如图 4 - 37 所示)、Adaptive Filters(自适应滤波器)子模块集(如图 4 - 38 所示)和 Multirate Filters(多级滤波器)子模块集(如图 4 - 39 所示)。

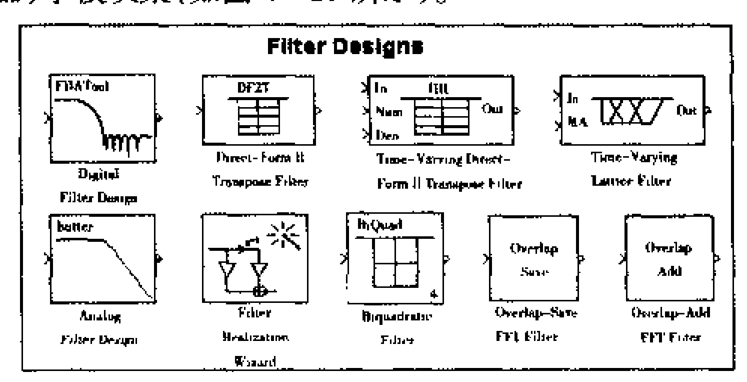


图 4 - 37　滤波器设计子模块集

在滤波器设计子模块集中包含了数字滤波器设计模块、模拟滤波器设计模块、时变格型滤波器设计模块以及滤波器实现仿真模块等各种滤波器设计的仿真模块。自适应滤波

器子模块集中包含了 LMS 自适应滤波器、Kalman 自适应滤波器和 RLS 自适应滤波器的设计仿真模块；多级滤波器子模块集中包含几种多级滤波器设计仿真模块。

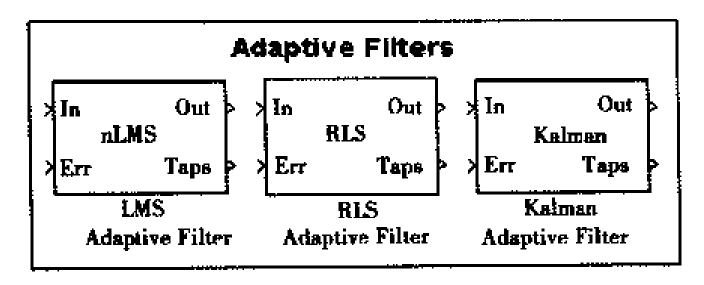


图 4－38　自适应滤波器子模块集

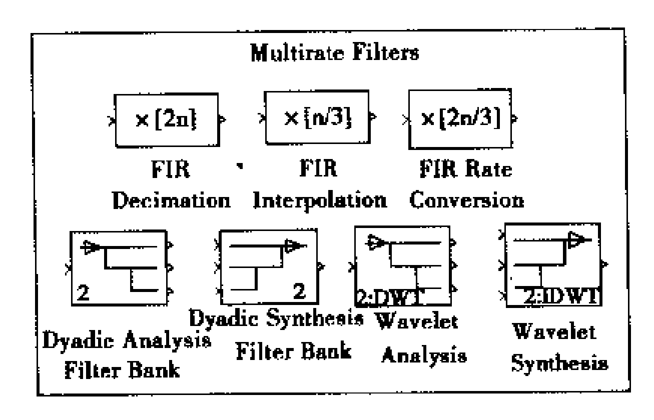


图 4－39　多级滤波器子模块集

◆ **Transforms（信号变换）子模块集**

Transforms(信号变换)子模块集中包含了快速傅立叶变换及其逆变换等模块，如图 4－40所示。

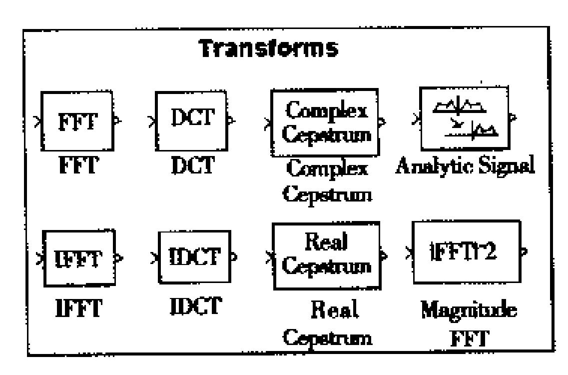


图 4－40　信号变换子模块集

◆ **其他子模块集**

DSP 模块库除了包含以上介绍的子模块集之外，还包含 Estimation(参数估计和功率谱估计)子模块集、Quantizers(量化)子模块集、Math Functions(数学函数)子模块集以及 Statistics(统计)子模块集等众多模块集，由于篇幅有限，在此不再作详细介绍。

4.7 本章小结

本章主要介绍了MATLAB在信号处理方面的应用。介绍了在信号处理中的各种常用输入信号，以使读者对信号处理中的信号源有一些了解；介绍了信号处理中非常重要的两种时频变换——Z变换和傅立叶变换，为进一步学习本章和本书以后章节的内容打下基础。数字滤波器的设计是信号处理的重要内容，本章用大量篇幅详细讲解了数字滤波器的结构、设计方法、MATLAB信号处理工具箱中的各种滤波器设计函数，并通过滤波器设计实例介绍了数字滤波器设计的具体方法与技巧。

习　题

1. 用MATLAB绘制下列各信号的波形图。

   (1) $te^{-t}U(t)$　　(2) $\sin 2\pi f_1 t+\cos 2\pi f_2 t$

   (3) $x(n)=(1/2)^{n-1}U(n)$　　(4) $x(n)=0.5n+\cos(5\pi/7)n$

   (5) $h(n)=3\delta(n)-\delta(n-1)$

   其中，$f_1=50\text{Hz}$，$f_2=200\text{Hz}$。

2. 用MATLAB函数产生一个矩形脉冲串，连续时间为3s，脉冲个数为15，并绘制其波形图。

3. 利用Z变换求解差分方程：

$$y(n)=(1/2)y(n-1)+x(n)$$

   其中，$x(n)=(0.5)^n U(n)$，$y(-1)=0.3$。

4. 用FFT绘制信号的幅频谱图。已知信号为：

$$x(t)=3\sin(2\pi t)+6\cos(10\pi t)+0.15w(t)$$

   其中，$w(t)$为白噪声。

5. 设计一个数字Chebyshev带通IIR滤波器，给定的指标为：

   (1) 波纹 $\delta_1\leqslant 2\text{dB}$，当 $200\text{Hz}\leqslant f\leqslant 400\text{Hz}$；

   (2) 衰减 $\delta_2\geqslant 20\text{dB}$，当 $f\leqslant 100\text{Hz}$，$f\geqslant 600\text{Hz}$；

   (3) 抽样频率 Fs = 2kHz。

   试分别用冲激响应不变法和双线性变换法进行设计，最后写出$H(z)$的表达式，并画出系统的幅频响应特性。

6. 设计一个数字Butterworth高通IIR滤波器，给定的指标为：

   (1) 衰减 $\delta_2\geqslant 30\text{dB}$，当 $f\leqslant 3\text{kHz}$；

   (2) 衰减 $\delta_1\leqslant 3\text{dB}$，当 $f\geqslant 5\text{kHz}$；

   (3) 抽样频率 Fs = 20kHz。

   试分别用冲激响应不变法和双线性变换法进行设计，最后写出$H(z)$的表达式，

并画出系统的幅频响应特性。

7. 试用矩形窗设计一个 FIR 线性相位低通数字滤波器，已知 $\omega c=0.5\pi, N-21$。求出 $h(n)$并画出对数频幅特性曲线。
8. 试用汉宁窗设计一个 FIR 线性相位低通数字滤波器，已知 $\omega c=0.4\pi, N=51$。求出 $h(n)$并画出对数频幅特性曲线。
9. 分别用函数 firls 和 remez 设计一个 50 阶多通带滤波器，滤波器理想幅频响应为：
   f=[0 0.1 0.15 0.25 0.3 0.4 0.45 0.55 0.6 0.7 0.75 0.85 01.9 1];
   m=[1 1 0 0 1 1 0 0 1 1 0 0 1 1];
   并绘制出理想滤波器和实际设计出的滤波器的幅频响应图。
10. 分别用函数 fir1 和 fir2 设计一个 24 阶 FIR 带通滤波器，通带频率 $0.35\pi\leqslant\omega\leqslant 0.65\pi$，并绘制出滤波器的幅频响应图。
11. 用窗函数法设计一个线性相位 FIR 低通数字滤波器，并满足如下性能指标：通带截止频率 $\omega p=0.5\pi$，阻带截止频率 $\omega s=0.66\pi$，阻带衰减不小于 40dB，通带波纹不大于 3dB。
12. 在 SPTool 环境下设计满足习题 11 要求的 FIR 数字滤波器。

第5章　MATLAB在通信系统仿真中的应用

知识点：

- 通信系统模型与仿真模型
- 通信系统仿真模块
- 通信系统仿真命令
- 通信系统仿真实例

本章主要介绍利用MATLAB进行通信系统仿真的两种方法：时间流仿真和数据流仿真。在时间流仿真中，主要介绍通信系统动态仿真的各种仿真模块；在数据流仿真中，主要介绍通信系统仿真的各种仿真函数。最后通过实例介绍这两种方法的具体运用。

通过本章的学习，读者可以了解如何使用MATLAB来进行各种通信系统的仿真，可以学会运用不同的方法进行通信系统仿真，并能掌握通信系统仿真的技巧。

在当今信息时代，通信技术的发展日新月异，计算机技术的发展突飞猛进，通信专业的学生和科技人员不但要掌握现代通信技术和理论，更需要了解和掌握基于计算机技术的通信系统仿真技术。

随着通信系统的复杂性不断增加，传统的设计方法已经不能适应发展的需要，因而通信系统的模拟仿真技术越来越受到工程技术人员的重视。传统的通信系统设计方法主要是手工分析与电路板试验，这些方法的最大缺点是比较繁杂，而且需要化很多时间。通信系统模拟环境可以称为之软件试验板，它可以使用户在很短的时间内建立整个通信系统模型，并对它进行模拟仿真，计算机通信系统模拟仿真环境是介于手工分析与电路试验板之间的一种通信系统设计方法。

MATLAB通信工具箱是一套用于通信领域理论研究、系统开发、分析设计和仿真的专业化工具软件包。MATLAB通信工具箱由两大部分组成：通信系统功能函数库和SIMULINK通信系统仿真模型库。

5.1　通信系统模型与仿真模型

要学习和掌握通信系统仿真技术，首先应该学习和了解通信系统仿真的相关基础知识。而在这些基础知识中，最重要的是通信系统的模型。

5.1.1　通信系统模型

通信是指消息传递的全过程，即信息的传输与交换。通信的目的在于传递信息，完成

信息传递所需要的全部设备和传输媒介的总和称为通信系统。如果信道中传输的是模拟信号，所对应的通信系统为模拟通信系统，如果信道中传输的是数字信号，所对应的通信系统为数字通信系统。

最简单的通信系统模型由信源、信道和信宿三个基本部分组成，同时在信道中还存在着干扰源，如图 5－1 所示。

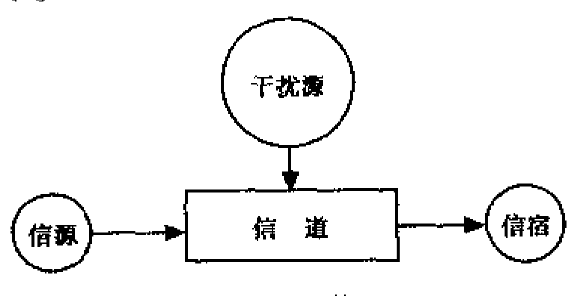


图 5－1　简单通信系统模型

实际的通信系统要比图 5－1 中的简单通信系统模型复杂得多，点对点通信系统的一般模型如图 5－2 所示，它反映了通信系统的共性。对图中各部分的作用简述如下：

- **信源**：其作用是把各种可能的消息转换成原始电信号，即非电/电转换。
- **发送设备**：用于将信源产生的消息信号变换成适合在信道中传输的信号，其变换过程包括编码和调制，其基本功能是将信源和信道匹配。
- **信道**：是信号传输的通道，即传输媒质，分为有线信道和无线信道两类。信道为信号提供了通道，同时也对信号产生各种干扰和噪声。
- **噪声源**：指信道中的噪声以及分散在通信系统其他各处的噪声的集中表示，它将影响通信质量。
- **接收设备**：它的功能与发送设备相反，它能从带有干扰的接收信号中正确恢复出相应的原始信号。
- **信宿**：其作用是将复原的原始电信号转换成消息，即电/非电转换。

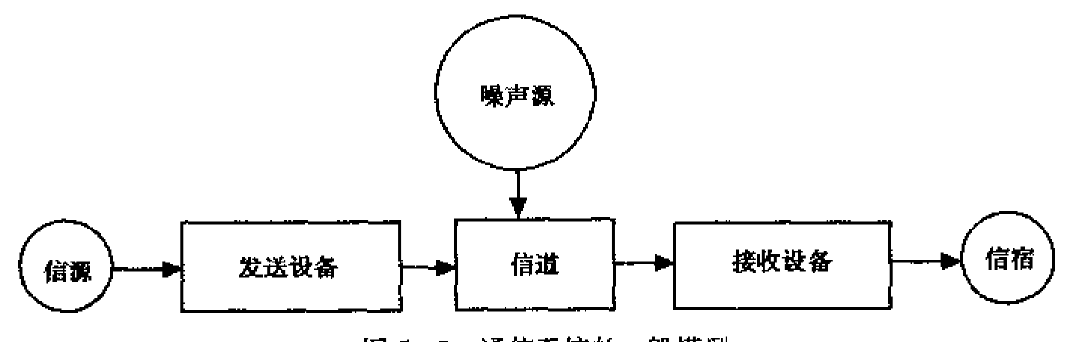


图 5－2　通信系统的一般模型

模拟通信系统应用比较早，也比较广泛，但近年来，数字通信系统以其显著的优越性得到了迅速的发展，而且日益兴旺，甚至有替代模拟通信系统的趋势。数字通信的基本特征是传输的信号是“离散”或“数字”的。数字通信系统就是利用数字信号来传递信息的通信系统。典型的数字通信系统的结构如图 5－3 所示。对图中各部分的作用和特点简述如下：

- 信源是产生消息的来源，消息可以是离散的，也可以是连续的。它可以是数据、

文字，也可以是语言、图像。

- 信源编码器的作用是将信源的输出变换为数字信息序列，信源编码的目的通常是降低信源输出中的多余度，减少每个消息、字符所需的平均码元数，从而提高信息传输或存储的有效性。
- 信道编码器的作用是对信源编码器的输出进行变换，用增加多余度的方法提高对信道干扰的抗击能力。
- 调制器的作用是将信道编码器输出的数字序列变换为振幅、频率或相位受到调节控制的波形，以适合在信道中作较长距离的传输。
- 信道是信号由发送端传输到接收端的媒介。典型的传输信道有明线、电缆、高频无线通道、微波通道和光导纤维通道等。
- 噪声源是对传输信道或存储介质构成干扰的来源的总称。在实际信道中，存在着各种各样的噪声。
- 解调器的作用是将从信道中传送过来的信号波形还原为调制以前的数字序列。
- 信道译码器的作用和信道编码器的作用相反，它利用信道编码时所提供的多余度，检查或者纠正解调器还原的数字序列中的错误，并把有用的信息序列送往信源译码器。
- 信源译码器的作用与信源编码器的作用相反，它把经过信道译码器核对后的信息序列转换为适合收信者接收的消息形式。
- 信宿就是消息要送往的目的地，即收信者。

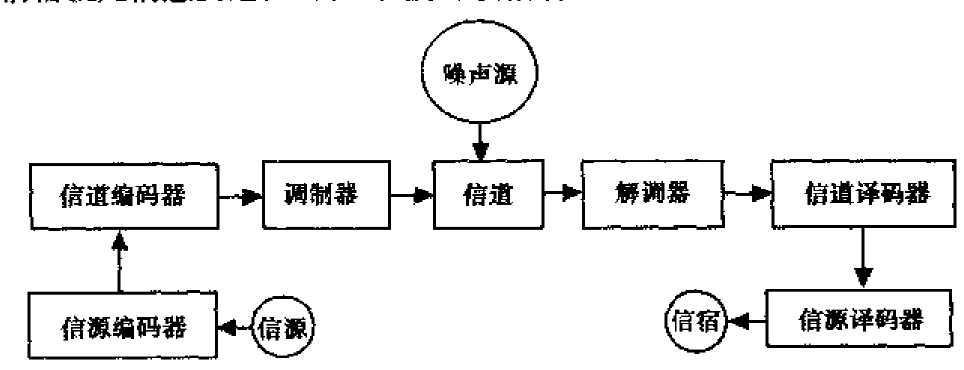


图 5－3　数字通信系统的一般模型已经修改

通信的任务是快速、准确地传递信息，从消息的传输方面来说，通信的有效性和可靠性是通信系统最主要的性能指标。有效性是指在给定信道内所传输的信息内容的多少，主要指消息传输的“速度”问题；可靠性是指接收信息的准确程度，主要指消息传输的“质量”问题，这两者是相互矛盾而又相互联系的。衡量数字通信系统的有效性的主要性能指标是传输速率、频带利用率；可靠性指标主要是差错率。

5.1.2　通信系统仿真模型

通信系统一般都可以建立数学模型。根据所需仿真的通信系统的数学模型，只要从各个模块集中找出所需的模块，将其用鼠标拖到模型窗口中组合在一起，并设定好各个模块参数就可方便地进行动态仿真。从输出模块可实时地查看仿真结果，如时域波形图、频

谱图等。每次仿真结束后还可以更改各个参数,以便观察仿真结果的变化情况。另外,对SIMULINK中没有的模块,可以自己创建所需的子模块,并且对其加以封装,以便随时调用。

在SIMULINK中,通信系统仿真的一般模型如图5-4所示。在图5-4中,每个框图都由一个子模块集构成。在通信系统中,一般情况下,传输和接收所采用的技术是相互对应的。在建立通信系统仿真模型时,只需从相应的子模块集中找出所需的模块,然后进行组合即可。下面通过一个简单实例加以说明。

提示:在SIMULINK环境下,建立通信系统的仿真模型时,所使用的仿真模块往往不限于上面所介绍的通信系统工具箱中的模块,而是要将多个子模块库结合起来使用,比如将Simulink(基本仿真子模块库)、DSP Blockset(数字信号处理子模块库)和Communication Blockset(通信系统子模块库)结合起来,才能完成各种通信系统的仿真。

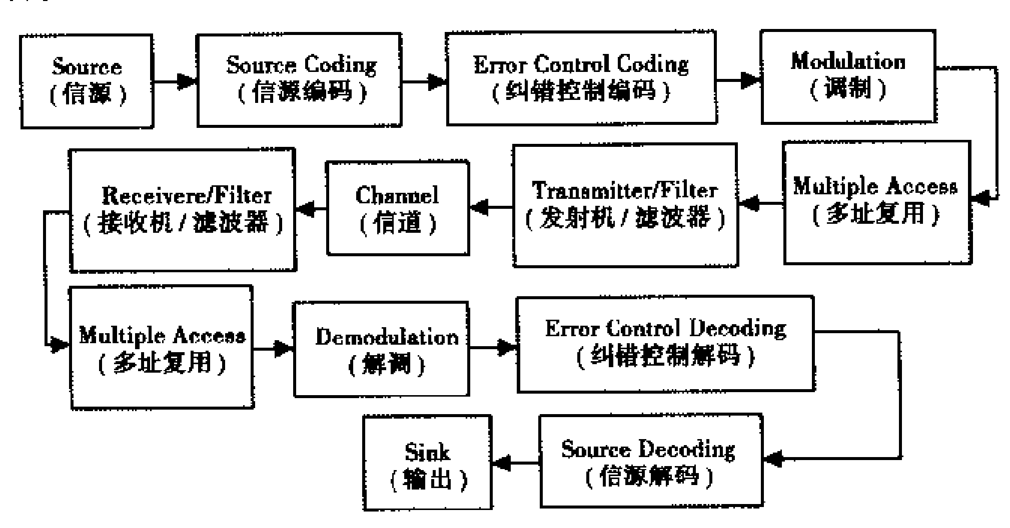


图5-4 通信系统仿真模型

例5-1 在时间域中观察有、无噪声情况下的正弦曲线。

分析:要在时间域中观察有、无噪声情况下的正弦曲线,首先应确定所需要的仿真模块:正弦信号产生器、噪声源、示波器;然后打开仿真模块库,在DSP Blockset中找到正弦波模块和向量示波器,并在Communication Blockset中找到噪声源模块;最后将它们连接起来,再设置各仿真模块的参数,并保存文件。通过以上步骤之后,所需的仿真模型已经建立起来,如图5-5所示,下面只需运行仿真即可。

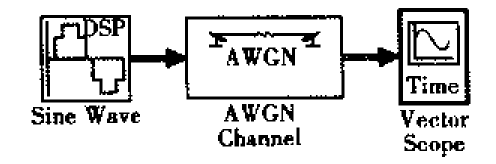


图5-5 例5-1的仿真模型

在图5-5的模型中,Sine Wave模块的参数设置为:Amplitude=5,Frenquency=30,Sample time=1/1000,Samples per frame=100;AWGN的参数设置为:Initial seed=1,Es/No(dB)=30;其余参数采用默认值。无噪声和有噪声的正弦波波形如图5-6和图5-7所示。

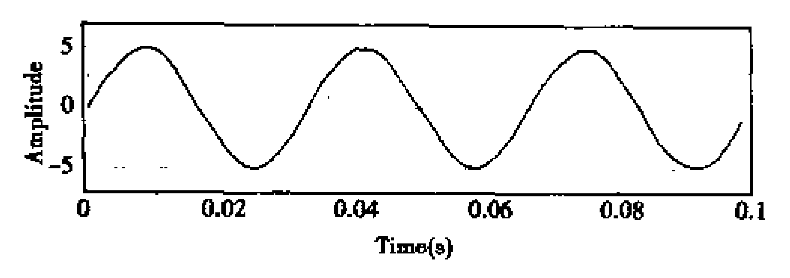


图 5-6　无噪声时的正弦波

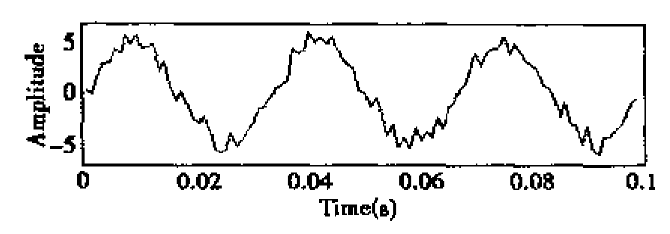


图 5-7　有噪声时的正弦波

5.2　通信系统仿真模块

在 MATLAB 中,用于通信系统仿真的模块主要由两大部分构成:通用仿真模块和专用仿真模块。它们分别属于标准仿真子模块库和通信子模块库。其中,Simulink(基本仿真子模块库)包含 Sources(信号源子模块集)、Sinks(信号接收子模块集)、Continuous(连续子模块集)、Discrete(离散子模块集)、Math(数学运算于模块集)、Functions & Tables(函数与表格子模块集)、Nonlinear(非线性子模块集)、Signals & Systems(信号与系统子模块集)、Subsystems(子系统子模块集)等 9 个子模块集。在 MATLAB 命令窗口中键入 commlibv2 命令,回车后,便可进入通信子模块库。在 Communications Blockset(通信子模块库)中提供了用于通信系统动态仿真所需的 10 个子模块集,分别是:Comm Sources(信源)、Source Coding(信源编码/译码)、Channel Coding(信道编码/译码)、Channels(信道)、Comm Sinks(信号接收)、Modulation(调制/解调)、Synchronization(同步)、Interleaving(去交错)、Basic Comm Functions(基本通信函数)和 Utility Functions(实用函数)子模块集,如图 5-8 所示。下面分别对各个子模块集进行介绍。

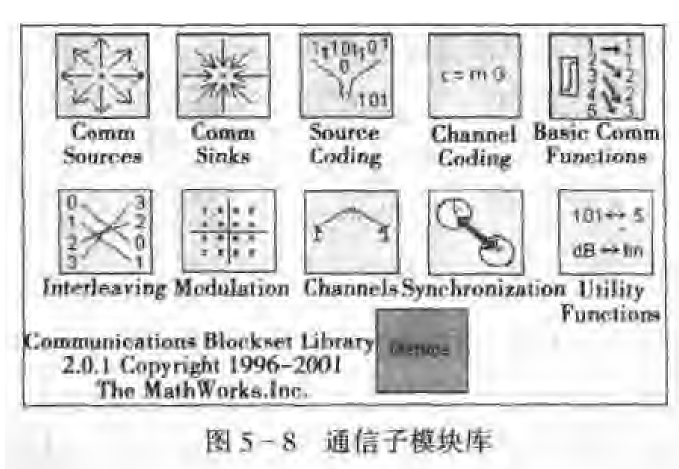


图 5-8　通信子模块库

5.2.1　Comm Sources(信源)子模块集

Comm Sources(信源)子模块集包含产生12种通信信号的仿真模块,如图5-9所示。下面分别对各个子模块进行介绍。

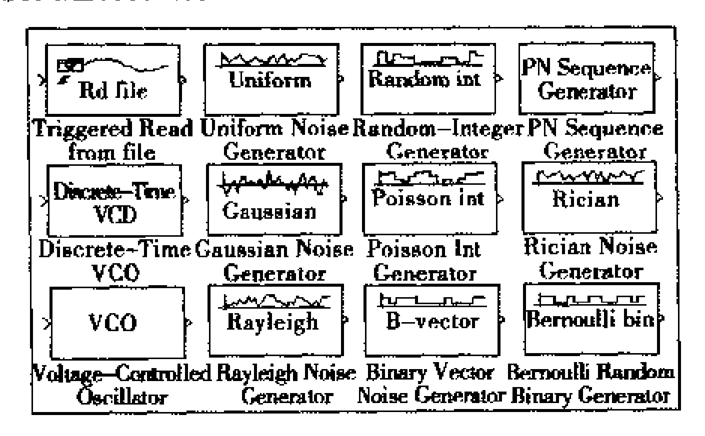


图5-9　信号源子模块集

◆　**Triggered Read from File(触发读取文件)模块**

Triggered Read from File(触发读取文件)模块仅在输入信号的上升沿才从文件中读取记录,读取的文件可以是ASCII码文件、数据文件或二进制文件。

◆　**Uniform Noise Generator(均匀分布随机噪声产生器)**

Uniform Noise Generator(均匀分布随机噪声产生器)将产生均匀分布的随机噪声,使用时应首先定义随机数种子。

◆　**Random-Integer Generator(均匀分布随机整数产生器)**

Random-Integer Generator(均匀分布随机整数产生器)用于产生均匀分布的随机整数,随机整数的范围为[0,M-1],其中M为指定的维数。

◆　**PN Sequence Generator(伪随机序列产生器)**

PN Sequence Generator(伪随机序列产生器)模块将产生伪噪声(pseudonoise)序列。

◆　**Discrete-Time VCO(离散时间压控振荡器)**

Discrete-Time VCO(离散时间压控振荡器)将产生一个频率随着输入信号幅值变化而变化的离散时间信号,其输入信号必须为标量。

◆　**Gaussian Noise Generator(高斯随机噪声产生器)**

Gaussian Noise Generator(高斯随机噪声产生器)将根据给定的均值和方差产生高斯分布的随机白噪声。

◆　**Poission Int Generator(泊松随机整数产生器)**

Poission Int Generator(泊松随机整数产生器)用于产生具有泊松分布的随机整数。

◆　**Rician Noise Generator(广义瑞利机噪声产生器)**

Rician Noise Generator(广义瑞利机噪声产生器)模块用于产生具有广义瑞利分布的随机噪声,输出向量的维数与参数设置中的随机种子的维数相同。

◆　**Voltage-Controlled Oscillator(VCO,压控振荡器)**

VCO模块将产生一个频率随着输入信号幅值变化而变化的连续时间信号,其输入信

号必须为标量。

◆ **Rayleigh Noise Generator(瑞利噪声产生器)**

Rayleigh Noise Generator(瑞利噪声产生器)用于产生具有瑞利分布的随机噪声,输出向量的维数与参数设置中的随机种子的维数相同。

◆ **Binay Vector Noise Generator(二进制向量噪声产生器)**

Binay Vector Noise Generator(二进制向量噪声产生器)用于产生二进制向量随机噪声。

◆ **Bernoulli Random Binary Generator(伯努利随机二进制信号产生器)**

Bernoulli Random Binary Generator(伯努利随机二进制信号产生器)用于产生服从伯努利分布的随机二进制数,模块输出向量的大小与参数设置中'0'出现概率大小的向量相同。

5.2.2 Source Coding(信源编/译码)子模块集

Source Coding(信源编/译码)子模块集提供了对信源进行编码和译码的仿真模块,主要包括信号量化、Differential Pulse Code Modulation(差分脉码调制)和压扩的仿真模块,如图 5-10 所示。下面分别对各个子模块进行介绍。

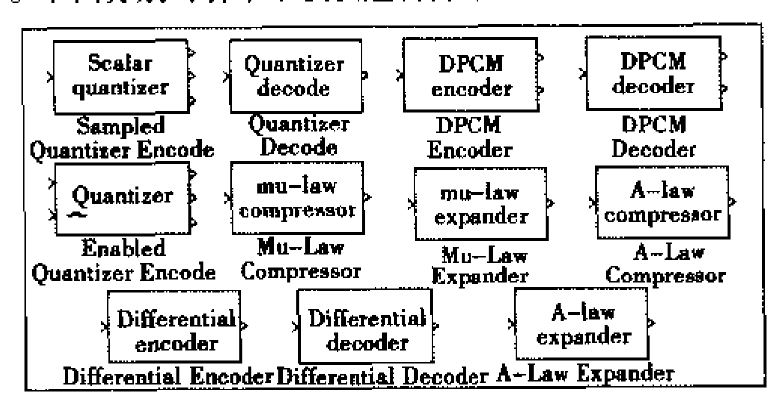


图 5-10　信源编/译码子模块集

◆ **Sampled Quantizer Encode(标量量化编码)模块**

Sampled Quantizer Encode(标量量化编码)模块采用标量量化法来量化消息信号。

◆ **Quantizer Decode(量化译码)模块**

Quantizer Decode(量化译码)模块的作用是从量化信号中恢复出消息信号。

◆ **DPCM Encoder(DPCM 编码器)**

DPCM Encoder(差分脉码调制编码器)用于对输入信号进行差分脉码调制。其原理是采用一个预测器并根据系统中以前的信号来估计下一个可能的值。

◆ **DPCM Decoder(DPCM 译码器)**

DPCM Decoder 用于将输入信号是采用 DPCM 编码方式产生的信号恢复为编码前的信号。该模块有两个输出端口,一个为恢复后的信号,另一个为预测误差。

◆ **Enabled Quantizer Encode(触发量化编码)模块**

Enabled Quantizer Encode(触发量化编码)模块的作用与标量量化编码模块类似,不同的只是其量化过程受它的第二个输入信号的控制。

◆ **Mu-Law Compressor(μ 律压缩器)**

μ 律压缩器采用 μ 律对数压缩计算对信号进行压缩。

◆ **Mu-Law Expander(μ 律扩张器)**

μ 律扩张是 μ 律压缩的逆运算,用于恢复采用 μ 律压缩器压缩的信号。

◆ **A-Law Compressor(A 律压缩器)**

A 律压缩器采用 A 律对数压缩计算对信号进行压缩。

◆ **A-Law Expander(A 律扩张器)**

A 律扩张是 A 律压缩的逆运算,用于恢复采用 A 律压缩器压缩的信号。

◆ **Differential Encoder(差分编码器)**

差分编码器的输出为一二进制标量,该输出是输入信号的逻辑差分运算。

◆ **Differential Decoder(差分译码器)**

差分译码是差分编码的逆运算,它将经过差分编码的信号恢复为编码前的信号。

5.2.3 Channel Coding(信道编/译码)子模块集

Channel Coding(信道编/译码)子模块集由 Block(分组码)子模块集和 Convolutional (卷积码)子模块集两部分构成。卷积码子模块集如图 5-11 所示,由 Convolutional Encoder(卷积编码器)、Viterbi Decoder(Viterbi 译码器)和 APP Decoder(APP 译码器)组成。

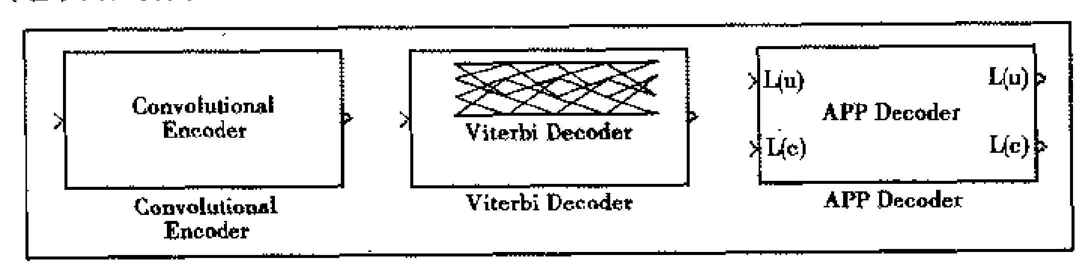


图 5-11　卷积码子模块集

分组码子模块集如图 5-12 所示。该模块集中的模块都是成对出现的,使用时也必须成对使用。各仿真模块组的功能如下:

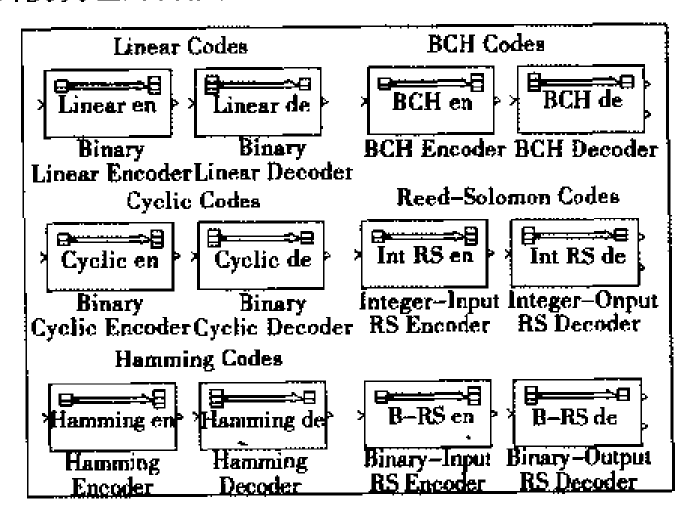


图 5-12　分组码子模块集

◆ **Linear Codes(线性码)模块组**

线性码模块组包括 Binary Linear Encoder(二进制线性编码器)和 Binary Linear Decoder(二进制线性译码器)。二进制线性编码器用于产生码元长度为 N、信息元长度为 K 的二进制线性分组码,这里的 N 和 K 分别表示线性分组码生成矩阵的列和行;二进制线性译码器用于将二进制线性分组码还原为消息。

◆ **BCH Codes(BCH 码)模块组**

BCH(Bose-Chaudhuri-Hocquenghem)Codes 模块组包括 BCH Encoder(BCH 编码器)和 BCH Decoder(BCH 译码器)。BCH 编码器用于产生码元长度为 N、信息元长度为 K 的 BCH 码,其中 $N=2^M-1$,M 为大于等于 3 的整数,K 的有效值是通过函数 bchpoly 计算得来的;BCH 译码器用于将 BCH 码还原为消息。

◆ **Cyclic Codes(循环码)模块组**

Cyclic Codes(循环码)模块组包括 Binary Cyclic Encoder(二进制循环编码器)和 Binary Cyclic Decoder(二进制循环译码器)。二进制循环编码器用于产生码元长度为 N、信息元长度为 K 的系统循环码,其中 $N=2^M-1$,M 为大于等于 3 的整数;二进制循环译码器用于将二进制循环码还原为消息。

◆ **Hamming Codes(海明码)模块组**

Hamming Codes(海明码)模块组包含 Hamming Encoder(海明编码器)和 Hamming Decoder(海明译码器)。海明编码器用于产生码元长度为 N、信息元长度为 K 的海明码,其中 $N=2^M-1$,M 为大于等于 3 的整数,K 必须等于N-M;海明译码器用于将海明码还原为消息。

◆ **Reed-Solomon Codes(R-S 码)模块组**

R-S 码模块组包含 Integer-Input RS Encoder(整数输入 R-S 编码器)、Integer-Output RS Decoder(整数输出 R-S 译码器)、Binary-Input RS Encoder(二进制输入 R-S 编码器)和 Binary-Output RS Decoder(二进制输出 R-S 译码器)。整数输入 R-S 编码器和二进制输入 R-S 编码器都用于产生码元长度为 N、信息元长度为 K 的 RS 码,其中 $N=2^M-1$,M 为大于等于 3 的整数;整数输出 R-S 译码器和二进制输出 R-S 译码器用于将相应的 RS 码还原为消息。

5.2.4 Channels(信道)子模块集

Channels(信道)子模块集包含四种不同的信道模块,如图 5-13 所示。下面分别对各模块进行介绍。

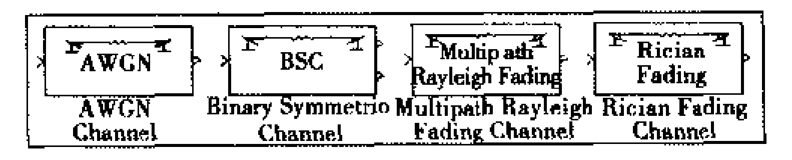


图 5-13 信道子模块集

◆ **AWGN Channel(加性高斯白噪声信道)**

AWGN Channel(加性高斯白噪声信道)是最普通、最常用的信道模型。该信道将高

斯白噪声作为噪声源，模块的输入和输出可以是实数或复数，此模块同时支持多通道输入输出。

◆ **Binary Symmetric Channel(二进制对称信道)**

二进制对称信道加入二进制误差作为噪声，误差概率可以是标量或与输入向量有相同大小的向量。

◆ **Multipath Rayleigh Fading Channel(多通道瑞利衰减信道)**

Multipath Rayleigh Fading Channel(多通道瑞利衰减信道)用于复数基带信号传输，它将输入信号与服从瑞利分布的复数随机噪声相乘，信道的数目与指定的延迟向量或增益向量的长度相等。

◆ **Rician Fading Channel(广义瑞利衰减信道)**

Rician Fading Channel(广义瑞利衰减信道)用于复数基带信号传输，它将输入信号与服从广义瑞利分布的复数随机噪声相乘。

5.2.5 Comm Sinks(信号接收)子模块集

Comm Sinks(信号接收)子模块集包含四种信号接收模块，如图 5－14 所示。下面分别对各模块进行介绍。

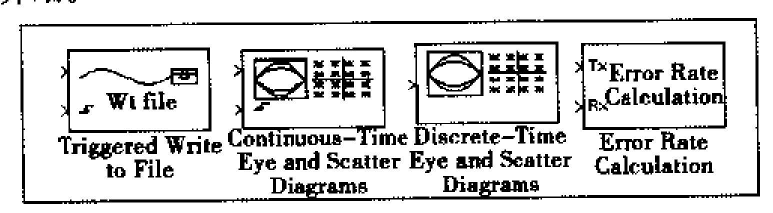


图 5－14　信号接收子模块集

◆ **Triggered Write to File(触发写文件)模块**

Triggered Write to File(触发写文件)模块有两个输入端，第一个输入端为待记录信号，第二个输入端为触发信号。此模块只有在触发信号的上升沿时，才执行将数据写入指定文件的操作。

◆ **Continuous-Time Eye and Scatter Diagrams(连续时间眼图和散布图)模块**

Continuous-Time Eye and Scatter Diagrams(连续时间眼图和散布图)模块用于根据不同的参数设置情况绘制出眼图、散布图、X-Y 坐标图，或者同时绘出眼图和 X-Y 坐标图、眼图和散布图。如果需要绘出散布图，则散布图的初始时间由触发输入端信号决定。

◆ **Discrete-Time Eye and Scatter Diagrams(离散时间眼图和散布图)模块**

Discrete-Time Eye and Scatter Diagrams(离散时间眼图和散布图)模块用于根据不同的参数设置情况绘制出眼图、散布图，或者同时绘出眼图和散布图。

◆ **Error Rate Calculate(误差率计算)模块**

Error Rate Calculate(误差率计算)模块用于计算接收数据的错误率。它有两个输入端口，一个接收发送方的输入信号，另一个接收接收方的输入信号。通过比较两个输入信号计算出误码率。

5.2.6 Modulation(调制/解调)子模块集

Modulation(调制/解调)子模块集又包含4个子模块集:Digital Baseband(数字基带)调制子模块集、Digital Passband(数字通带)调制子模块集、Analog Baseband(模拟基带)调制子模块集和 Analog Passband(模拟通带)调制子模块集。

1. 数字基带调制子模块集

Digital Baseband(数字基带调制)子模块集又包含4个子模块组:AM(幅度调制)子模块组、PM(相位调制)子模块组、FM(频率调制)子模块组和 CPM(连续相位调制)子模块组。下面分别对各子模块组进行介绍。

◆ **AM(幅度调制)子模块组**

AM(幅度调制)子模块组包含3种调制/解调模块:M-PAM Modulator Baseband and M-PAM Demodulator Baseband(M-PAM 调制和解调)模块、Rectangular QAM Modulator Baseband and Rectangular QAM Demodulator Baseband(矩形 QAM 调制和解调)模块、General QAM Modulator Baseband and General QAM Demodulator Baseband(通用 QAM 调制和解调)模块等。如图 5-15 所示。

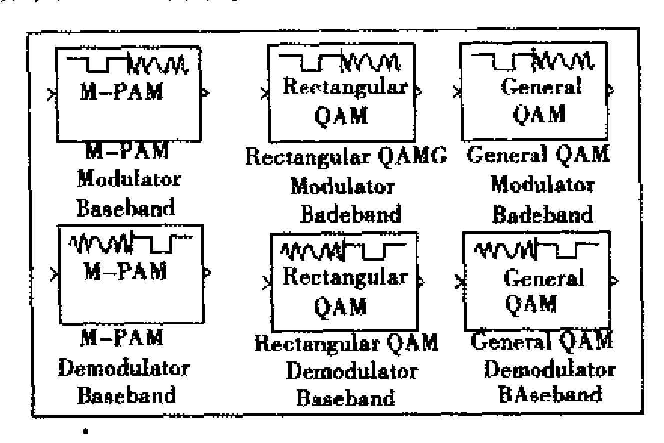


图 5-15 幅度调制(AM)子模块组

◆ **PM(相位调制)子模块组**

PM(相位)调制子模块组包括7种调制/解调模块:M-PSK Modulator Baseband and M-PSK Demodulator Baseband (M-PSK 调制/解调)模块、M-DPSK Modulator Baseband and M-DPSK Demodulator Baseband (M-DPSK 调制/解调)模块、BPSK Modulator Baseband and BPSK Demodulator Baseband(BPSK 调制/解调)模块、DBPSK Modulator Baseband and DBPSK Demodulator Baseband (DBPSK 调制/解调)模块、QPSK Modulator Baseband and QPSK Demodulator Baseband (QPSK 调制/解调)模块、DQPSK Modulator Baseband and DQPSK Demodulator Baseband (DQPSK 调制/解调)模块和 OQPSK Modulator Baseband and OQPSK Demodulator Baseband(OQPSK 调制/解调)模块等。如图 5-16所示。

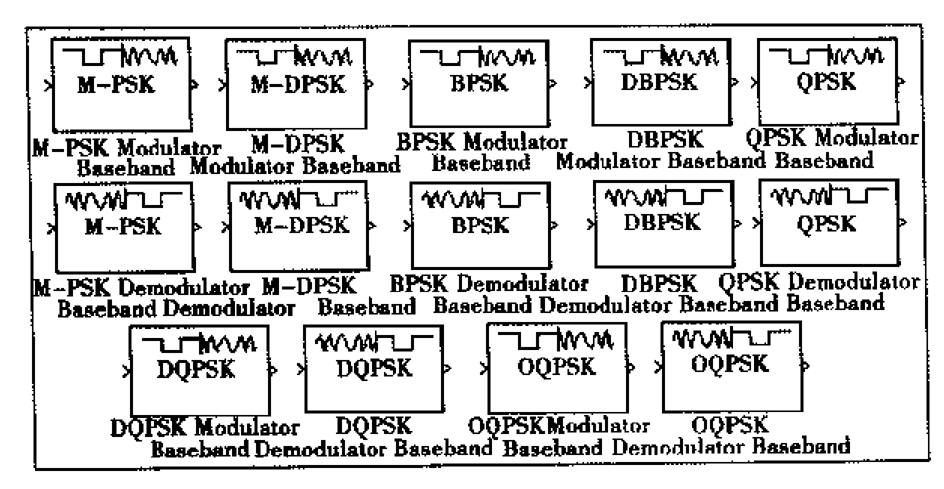


图 5－16　PM(相位)调制子模块组

◆　**FM(频率调制)子模块组**

FM(频率调制)子模块组中只包括 M-FSK 调制/解调模块。如图 5－17 所示。

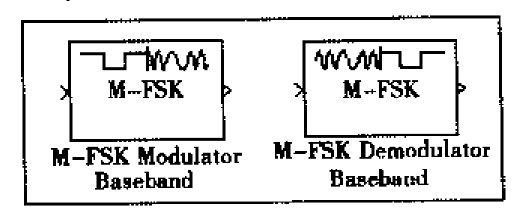


图 5－17　频率调制(FM)子模块组

◆　**CPM(连续相位调制)子模块组**

CPM(连续相位调制)子模块组包含 4 种调制/解调方式:CPM 调制/解调、GMSK 调制/解调、MSK 调制/解调、CPFSK 调制/解调。如图 5－18 所示。

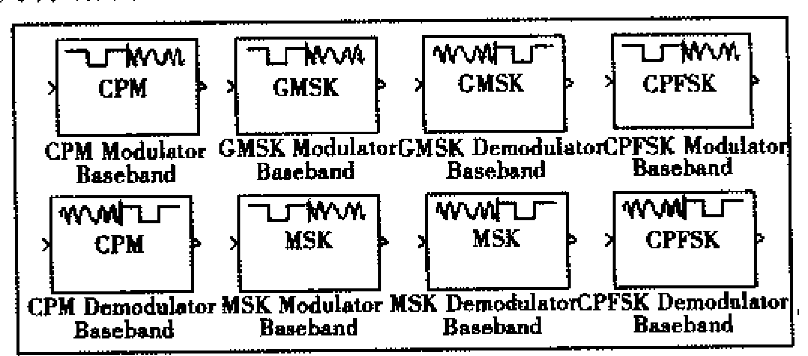


图 5－18　CPM 子模块组

2. 数字通带调制子模块集

该子模块集又包含 4 个子模块组:AM(幅度调制)子模块组、PM(相位调制)子模块组、FM(频率调制)子模块组和 CPM(连续相位调制)子模块组。各子模块组的详细情况如下:

◆　**AM(幅度调制)子模块组**

AM 子模块组包含 3 种调制/解调模块: M-PAM Modulator Passband and M-PAM

Demodulator Passband(M-PAM 调制和解调)模块、Rectangular QAM Modulator Passband and Rectangular QAM Demodulator Passband(矩形 QAM 调制和解调)模块和 General QAM Modulator Passband and General QAM Demodulator Passband(通用 QAM 调制和解调)模块。如图 5－19 所示。

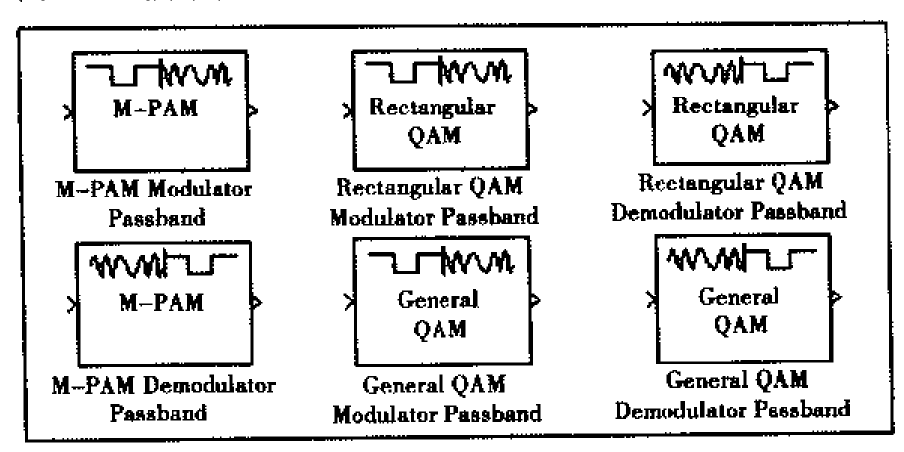


图 5－19　AM 子模块组

◆ **PM(相位调制)子模块组**

PM 子模块组有 3 种调制/解调模块：M-PSK 调制/解调模块、OQPSK 调制/解调模块和 M-DPSK 调制/解调模块。如图 5－20 所示。

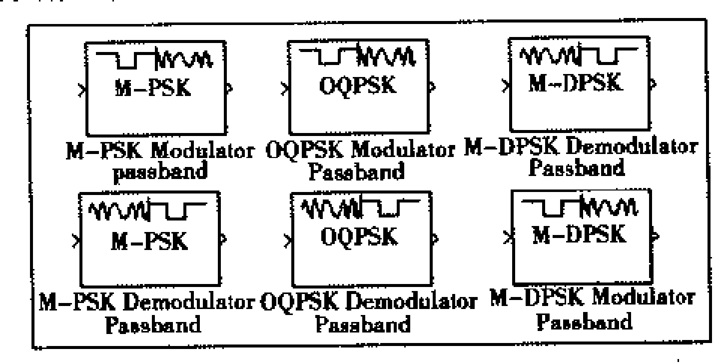


图 5－20　PM 子模块组

◆ **FM(频率调制)子模块组**

FM 子模块组只有 M-FSK 调制/解调模块。如图 5－21 所示。

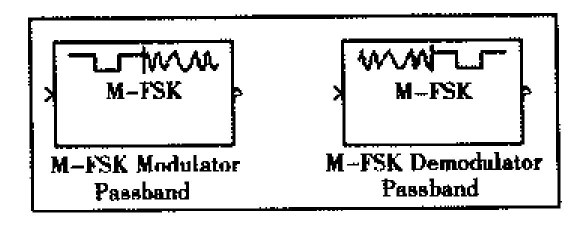


图 5－21　FM 子模块组

◆ **CPM 子模块组**

CPM 子模块组包含 4 种调制/解调方式：CPM 调制/解调、GMSK 调制/解调、MSK 调制/解调和 CPFSK 调制/解调。如图 5－22 所示。

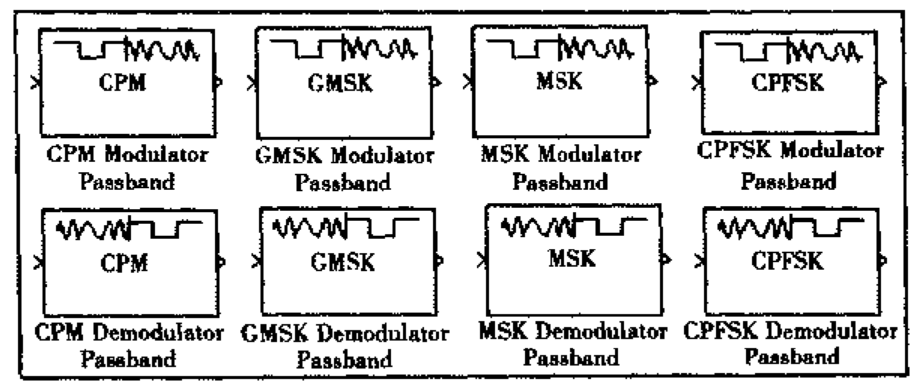


图5－22　CPM子模块组

3．模拟基带调制子模块集

模拟基带调制子模块集包含5种调制/解调方式：DSBSC AM Modulator Baseband and DSBSC AM Demodulator Baseband（抑制载波双边带振幅调制/解调）、FM（频率）调制/解调、PM（相位）调制/解调、SSB AM（单边带振幅）调制/解调和DSB AM（双边带振幅）调制/解调等，如图5－23所示。

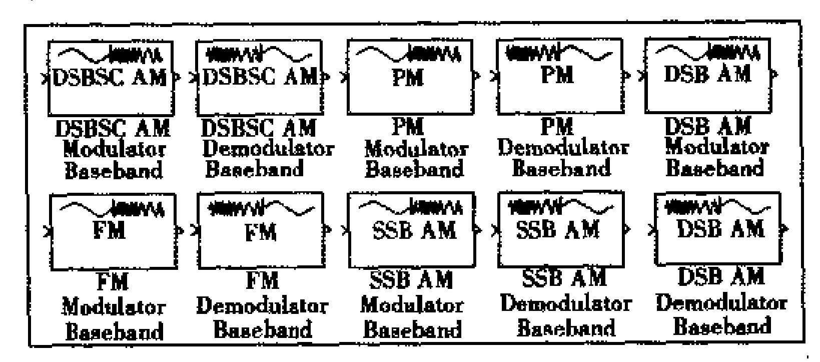


图5－23　模拟基带调制子模块集

4．模拟通带调制子模块集

模拟通带调制子模块集包含5种调制/解调方式：DSBSC AM Modulator Passband and DSBSC AM Demodulator Passband（抑制载波双边带振幅）调制/解调、FM（频率）调制/解调、PM（相位）调制/解调、SSB AM（单边带振幅）调制/解调和DSB AM（双边带振幅）调制/解调，如图5－24所示。

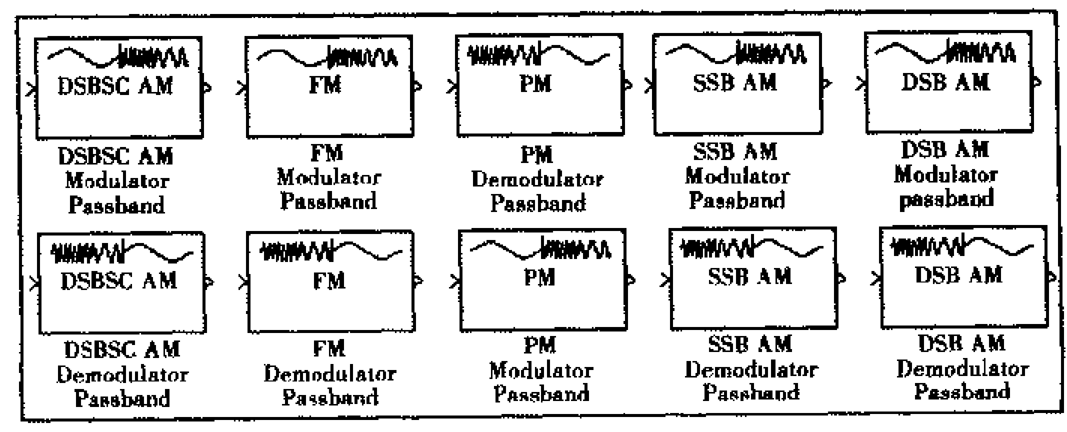


图5－24　模拟通带调制子模块集

5.2.7 Synchronization(同步)子模块集

Synchronization(同步)子模块集包含有 4 个模块,如图 5-25 所示。下面分别对它们进行介绍。

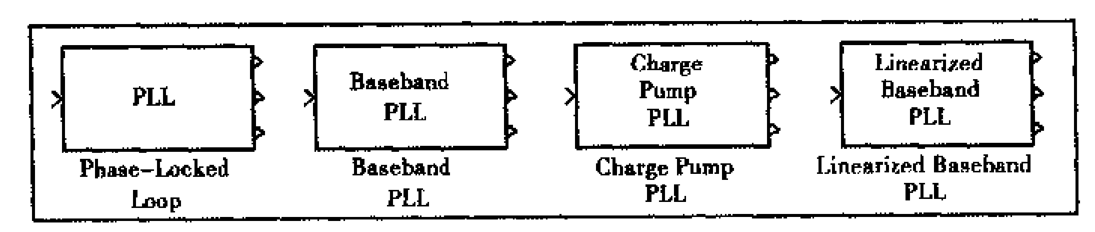


图 5-25　同步子模块集

◆ **Phase-Locked Loop(PLL,锁相环)模块**

PLL 模块能够自动修正本地信号的相位来匹配输入信号的相位。此模块是一个相位匹配模块,使用的是一个简单的锁相环。它有三个输出端口:低通滤波器的输出、检测出的相位偏差输出和压控振荡器 VCO 的输出等端口。

◆ **Baseband PLL(基带锁相环)模块**

Baseband PLL(基带锁相环)模块采用的是基带模型方式的相同步模块,它等效于基带模型上的 PLL。该模块有三个输出端口,分别是低通滤波器的输出、检测出的相位偏差输出和压控振荡器 VCO 的输出等端口。

◆ **Charge Pump PLL(进料泵锁相环)模块**

Charge Pump PLL(进料泵锁相环)模块采用的是数字相位检测器,同时它也可以检测出频率,因而也称为相位/频率检测器。它也有三个输出端口:低通滤波器的输出、检测出的相位偏差输出和压控振荡器 VCO 的输出等端口。

◆ **Linearized Baseband PLL(线性化基带锁相环)模块**

Linearized Baseband PLL(线性化基带锁相环)模块采用线性基带模型方式实现相同步。它也有三个输出端口:低通滤波器的输出、检测出的相位偏差输出和压控振荡器 VCO 的输出等端口。

5.2.8 Interleaving(交错/去除交错)子模块集

交错是减少因信道有突发噪声而导致的比特错误率的常用技术。采用交错技术后,码元在传输到信道之前先采用交错技术,将码元的顺序重排,在接收端再采用去交错技术恢复码元先前的顺序。交错的目的是使信道中由于噪声导致的错误分布均匀,防止错误集中在某一码元。交错/去交错子模块集包含 Block(分组)交错和 Convolutional(卷积)交错两个子模块组,分别如图 5-26 和图 5-27 所示。

Block(分组)交错子模块组包含 Genetal Block Interleaver and Deinterleaver(通用分组交错器/去交错器)、Matrix Interleaver and Deinterleaver(矩阵交错器/去交错器)、Matrix Helical Scan Interleaver and Deinterleaver(矩阵螺旋扫描交错器/去交错器)、Algebraic In-

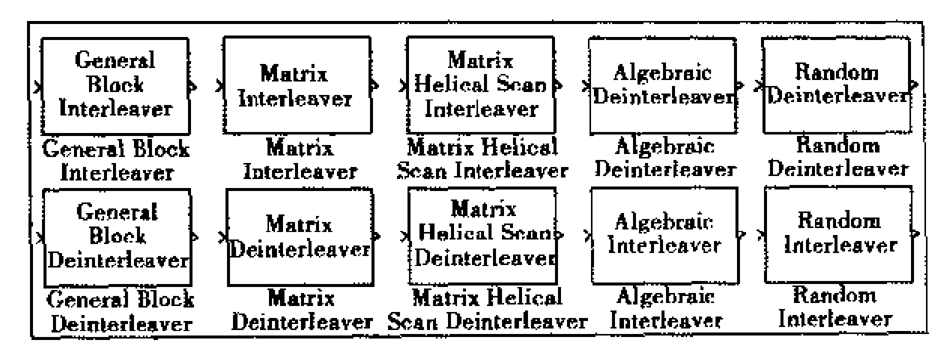


图5-26　分组交错子模块组

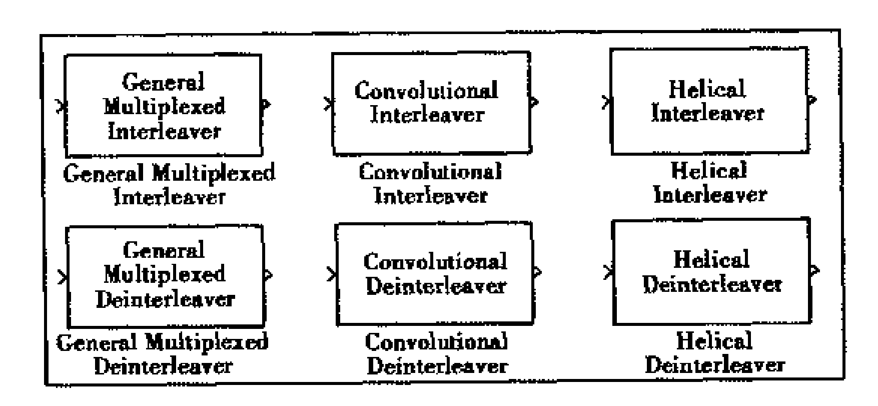


图5-27　卷积交错子模块组

terleaver and Deinterleaver(代数交错器/去交错器)以及 Random Interleaver and Deinterleaver(随机交错器/去交错器)等模块;Convolutional(卷积)子模块组包含 General Multiplexed Interleaver and Deinterleaver(通用多路交错器/去交错器)、Convolutional Interleaver and Deinterleaver(卷积交错器/去交错器)和 Helical Interleaver and Deinterleave(螺旋交错器/去交错器)等模块。

5.2.9　Basic Comm Functions(基本通信函数)子模块集

Basic Comm Functions(基本通信函数)子模块集又包含两个子模块组:Integrators(积分器)子模块组和 Sequence Operations(序列操作)子模块组。两个子模块组分别如图5-28和图5-29所示。因篇幅原因不作详细介绍。

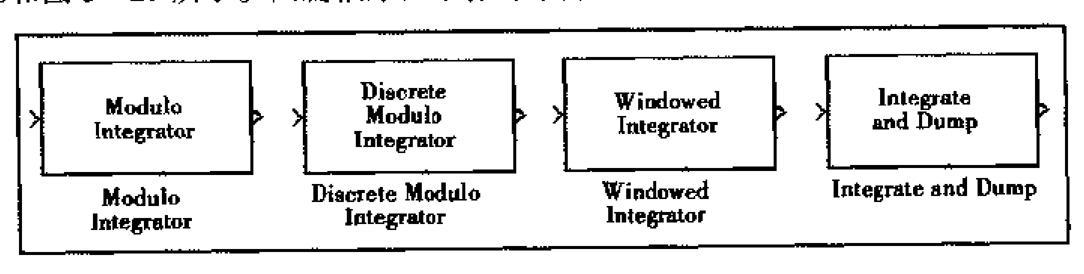


图5-28　积分器子模块组

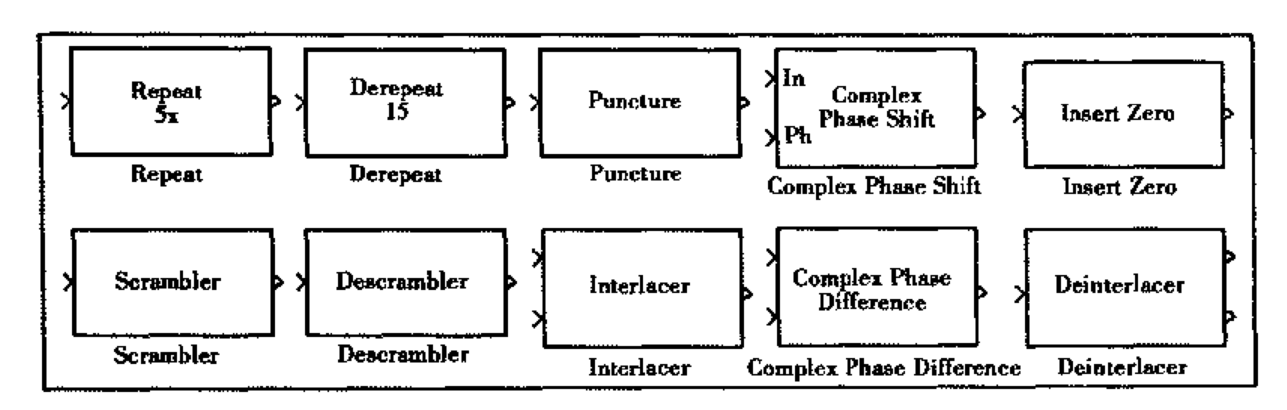


图 5-29　序列操作子模块组

5.2.10　Utility Functions(实用函数)子模块集

Utility Functions(实用函数)子模块集包含 6 种函数模块:Data Mapper(数据映射)模块、Bipolar to Unipolar Converter(双极向单极转换)模块、Integer to Bit Converter(整数转换成二进制表示)模块、Bit to Integer Converter(二进制转换成整数表示)模块、Unipolar to Bipolar Converter(单极向双极转换)模块和 dB Conversion(dB 转换)模块,如图 5-30 所示。

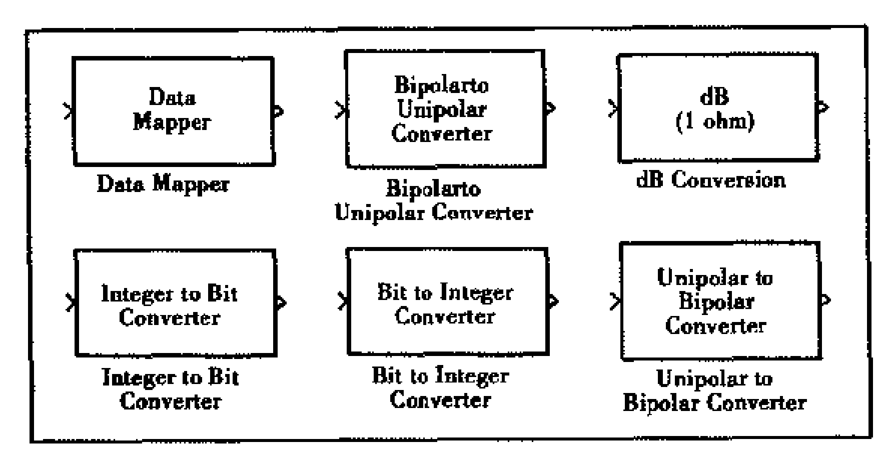


图 5-30　实用函数子模块集

5.3　通信系统仿真命令

MATLAB 通信系统工具箱提供了许多与通信系统有关的函数，其中包括信号源产生函数、信号分析函数、信源编码/解码函数、纠错控制编码/解码函数、纠错控制编码/解码底层函数、调制/解调函数（基带和通带）、特殊滤波器函数、特殊滤波器底层函数、传输信道模型函数、伽罗华域计算函数、实用工具函数等。下面分别对各个函数进行介绍。

5.3.1 信号源产生函数

MATLAB通信系统工具箱提供了5个信号源产生函数,如表5-1所示。

表5-1　信号源产生函数表

信号源产生函数	说　明
RANDERR	产生误比特数图样
RANDINT	产生均匀分布的随机整数矩阵
RANDSRC	产生指定符号的随机矩阵
WGN	产生高斯白噪声
AWGN	产生加性高斯白噪声

表中各函数的使用方法如下:

◆ **randerr 函数**

randerr函数在纠错编码中特别有用,它将产生误比特图样。其调用格式为:

OUT = RANDERR(M)

该函数格式用于产生一个M阶二进制方阵,且方阵的每一行恰好只有一个1。

OUT = RANDERR(M,N)

该函数格式用于产生一个M行N列二进制矩阵,且矩阵的每一行恰好只有一个1。

OUT = RANDERR(M,N,ERRORS)

该函数格式将产生一个M行N列二进制矩阵,参数ERRORS可以是一个标量、行向量或只有两行的矩阵。当ERRORS为一标量时,产生的矩阵的每一行中1的个数等于ERRORS;当ERRORS为一向量时,产生的矩阵的每一行中出现1的可能个数由向量ERRORS的相应元素指定,每一元素出现1的概率是相等的;当ERRORS为两行的矩阵时,第一行指定出现1的可能个数,第二行说明出现1的概率,第二行中所有元素的和应等于1。

OUT = RANDERR(M,N,ERRORS,STATE)

该函数格式可以重新设置其状态。

◆ **randint 函数**

randint函数可以产生均匀分布的随机整数矩阵。其调用格式为:

OUT = RANDINT

此函数格式将以等概率方式产生一个0或1。

OUT = RANDINT(M)

此函数格式将以等概率方式产生一个由0和1组成的M阶随机方阵。

OUT = RANDINT(M,N)

此函数格式将以等概率方式产生一个由0和1组成的M行N列随机矩阵。

OUT = RANDINT(M,N,RANGE)

此函数格式将产生一个M行N列随机整数矩阵,RABGE可以是标量,也可以是向量。

OUT = RANDINT(M,N,RANGE,STATE)

此函数格式可以重新设置其状态。

◆ **Randsrc 函数**

Randsrc 函数将按指定的符号产生随机矩阵。其调用格式为：

```
OUT = RANDSRC
OUT = RANDSRC(M)
OUT = RANDSRC(M,N)
OUT = RANDSRC(M,N,ALPHABET)
OUT = RANDSRC(M,N,ALPHABET,STATE)
```

Randsrc 函数产生的符号由参数 ALPHABET 指定，该参数可以是行向量或矩阵，如果省略，则以等概率产生 -1 和 1 的随机矩阵。

◆ **wgn 函数**

wgn 函数能产生高斯白噪声，其调用格式为：

```
Y = WGN (M,N,P)
Y = WGN(M,N,P,IMP)
Y = WGN(M,N,P,IMP,STATE)
Y = WGN(..., POWERTYPE)
Y = WGN(..., OUTPUTTYPE)
```

函数 wgn 用于产生 M 行 N 列的高斯白噪声矩阵。其中，参数 P 用于指定输出噪声的功率，P 的单位由 POWERTYPE 指定，可以是'dB'、'dBm'或'linear'；参数 IMP(欧姆)指定负载阻抗；参数 STATE 指定其状态；OUTPUTTYPE 为输出类型，可以是复数或实数。

◆ **awgn 函数**

awgn 函数将产生加性高斯白噪声。其调用格式为：

```
Y = AWGN(X,SNR)
Y = AWGN(X,SNR,SIGPOWER)
Y = AWGN(X,SNR,SIGPOWER,STATE)
Y = AWGN(..., POWERTYPE)
```

其中，X 为信号；SNR 为信噪比，单位为 dB；当 SIGPOWER 为数字时，信号功率的单位为 dB，当 SIGPOWER 为'measured'时，在加入噪声前要先测量信号的功率；POWERTYPE 用于指定 SNR 和 SIGPOWER 的单位，可以是'dB'或'linear'，'linear'代表瓦。

5.3.2 信号分析函数

MATLAB 通信系统工具箱提供了 4 个信号分析函数，如表 5-2 所示。

表 5-2 信号分析函数表

信号分析函数	说　明
BITERR	误比特数及误比特率计算
EYEDIAGRAM	产生眼图
SCATTERPLOT	产生散布图
SYMERR	误符号数和误符号率计算

表中各函数的使用方法如下：

◆ **biterr 函数**

biterr 函数用于计算两个输入矩阵 X 和 Y 的不同元素的个数和的比值。其调用格式为：

```
[NUMBER,RATIO] = BITERR(X,Y)
[NUMBER,RATIO] = BITERR(X,Y,K)
[NUMBER,RATIO] = BITERR(X,Y,K,FLAG)
[NUMBER,RATIO,INDIVIDUAL] = BITERR(X,Y,K,FLAG)
```

其中，参数 K 指定两矩阵中每个元素用 K 位二进制表示；FLAG 的值为'column-wise'，表示将两个输入矩阵 X 和 Y 按列进行比较，为'row-wise'表示将两个输入矩阵 X 和 Y 按行进行比较，为'overall'表示将两个输入矩阵 X 和 Y 的行和列均进行比较。

◆ **eyediagram 函数**

eyediagram 函数用于产生 N 点采样信号 X 的眼图。其调用格式为：

```
EYEDIAGRAM(X, N)
EYEDIAGRAM(X, N, PERIOD)
EYEDIAGRAM(X, N, PERIOD, OFFSET)
EYEDIAGRAM(X, N, PERIOD, OFFSET, PLOTSTRING)
```

其中，参数 PERIOD 指出眼图 X 坐标轴的范围，X 坐标为-PERIOD/2～PERIOD/2，其默认值为 1；OFFSET 指出眼图 X 坐标的中心，0≤OFFSET＜N；PLOTSTRING 指出图线的颜色和线型。

◆ **scatterplot 函数**

scatterplot 函数用于绘制出信号 X 的散布图。其调用格式为：

```
SCATTERPLOT(X)
SCATTERPLOT(X, N)
SCATTERPLOT(X, N, OFFSET)
SCATTERPLOT(X, N, OFFSET, PLOTSTRING)
```

其中，参数 N 为抽取因子，表示每隔 N 点的信号值将在图中绘出，默认值为 1；其余参数的含义请参见 eyediagram 函数。

◆ **symerr 函数**

symerr 函数用于比较两个矩阵 X 和 Y 的不同元素的个数和比率。其调用格式为：

```
[NUMBER,RATIO] = SYMERR(X,Y)
```

5.3.3 信源编码/解码函数

MATLAB 通信系统工具箱提供了 5 个信源编码/解码函数，如表 5－3 所示。

表 5－3　信源编码/解码函数表

信源编码/解码函数	说　明
COMPAND	M 律或 A 律压扩计算
DPCMDECO	差分脉码调制译码
DPCMENCO	差分脉码调制编码
LLOYDS	使用 LLOYD 算法优化量化参数
QUANTIZ	产生量化序号和量化值

表中各函数的作用和使用方法如下：

◆ **compand 函数**

在信源编码中，compand 函数按 μ 律或 A 律对输入信号进行扩展和压缩。其调用格式为：

OUT = COMPAND(IN, PARAM, V)

此函数格式用于完成指定输入信号的最大幅值 V 的 μ 律压缩，PARAM 指出 μ 或 A 的值。

OUT = COMPAND(IN, PARAM, V, METHOD)

此函数格式可实现 μ 律或 A 律的压缩和扩展，具体采用哪种方式由参数 METHOD 决定，METHOD 的取值及含义如表 5－4 所示。

表 5－4　METHOD 的选取含义

METHOD	含　义
'MU/COMPRESSOR'	M 律压缩
'MU/EXPANDER'	M 律扩展
'A/COMPRESSOR'	A 律压缩
'A/EXPANDER'	A 律扩展

◆ **dpcmdeco 函数**

dpcmdeco 函数用于实现差分脉码调制解调。其调用格式为：

SIG = DPCMDECO(INDX, CODEBOOK, PREDICTOR)

[SIG, QUANT] = DPCMDECO(INDX, CODEBOOK, PREDICTOR)

其中，CODEBOOK 为预测误差量化码本；PREDICTOR 为预测器的预测传递函数系数向量，通常 M 阶传递函数的表示形式为[0, n1, n2, ...nM]；INDX 为量化序号；QUANT 为量化的预测误差。

◆ **dpcmenco 函数**

dpcmenco 函数用于实现差分脉码调制编码。函数的格式为：

INDX = DPCMENCO(SIG, CODEBOOK, PARTITOIN, PREDICTOR)

[INDX, QUANT] = DPCMENCO(SIG, CODEBOOK, PARTITION, PREDICTOR)

其中，SIG 为输入信号；PARTITION 为量化阈值；其余参数的含义与 dpcmdeco 函数中的含义一样。

◆ **lloyds 函数**

lloyds 函数能够优化标量量化的阈值和码本。它使用 Lloyds_max 算法优化标量量化参数，用给定的训练序列矢量优化初始码本，使量化误差小于给定的容差。函数的调用格式为：

[PARTITION, CODEBOOK] = LLOYDS(TRAINING_SET, INI_CODEBOOK)

[PARTITION,CODEBOOK] = LLOYDS(TRAINING_SET, INI_CODEBOOK, TOL)

[PARTITION, CODEBOOK, DISTORTION] =

　　　　LLOYDS(TRAINING_SET, INI_CODEBOOK, TOL)

[PARTITION, CODEBOOK, DISTORTION, REL _ DISTORTION] =

LLOYDS(TRAINING _ SET, INI _ CODEBOOK, TOL)

其中,TRAINING _ SET 为给定的训练序列;INI _ CODEBOOK 为码本的初始预测值;TOL 为给定的容差;在输出值中,REL _ DISTORTION 为相对量化误差;PARTITION 为量化间隔。

◆ **quantiz 函数**

quantiz 函数用于产生一个量化序号和输出量化值。函数的调用格式为:

INDX = QUANTIZ(SIG, PARTITION)

[INDX, QUANT] = QUANTIZ(SIG, PARTITION, CODEBOOK)

[INDX, QUANT, DISTOR] = QUANTIZ(SIG, PARTITION, CODEBOOK)

其中,SIG 为输入信号;PARTITION 为量化间隔;CODEBOOK 为码本;INDX 为量化序号;QUANT 为量化误差;DISTOR 为量化的预测误差。

5.3.4 纠错控制编码/解码函数

MATLAB 通信系统工具箱提供了如表 5－5 所示的纠错控制编码/解码函数。表中各函数的含义和调用方法如下:

表 5－5　纠错控制编码/解码函数表

纠错控制编码/解码函数	说　明
BCHPOLY	产生二进制 BCH 码的参数或生成多项式
CONVENC	卷积码编码计算
CYCLGEN	产生循环码的生成矩阵和校验矩阵
CYCLPOLY	产生循环码的生成多项式
DECODE	分组码译码器
ENCODE	分组码编码器
GEN2PAR	生成矩阵和校验矩阵之间的转换
GFWEIGHT	计算线性分组码的最小码距
HAMMGEN	产生海明码的生成矩阵和校验矩阵
RSDECOF	对已经编码的 ASCII 文件用 R-S 码进行译码
RSENCOF	对 ASCII 文件进行 R-S 码编码
RSPOLY	产生 R-S 码的生成多项式
SYNDTABLE	产生综合译码表
VITDEC	用 VITERBI 算法进行卷积译码

◆ **bchpoly 函数**

bchpoly 函数用于产生二进制 BCH 码的参数和生成多项式。其调用格式为：

```
BCHPOLY
Pg = BCHPOLY
Pg = BCHPOLY(N)
Pg = BCHPOLY(N, K)
[Pg, Pm] = BCHPOLY(N, K)
[Pg, Pm, CS] = BCHPOLY(N, K)
[Pg, Pm, CS, H] = BCHPOLY(N, K)
[Pg, Pm, CS, H, T] = BCHPOLY(N, K)
```

bchpoly 函数有多种调用格式，用 BCHPOLY 可列出码长小于 511 时的信息元长度和纠错能力：Pg 的第一列为码字长度，第二列为信息元长度，第三列为纠错能力；Pm 为生成的最小多项式；CS 为 Pg 多项式的陪集；H 为校验阵；T 为纠错容量；调用该函数时，可指定码字长 N 和信息元长度 K。

◆ **convenc 函数**

convenc 函数用于卷积码编码计算。其调用格式为：

```
CODE = CONVENC(MSG,TRELLIS)
CODE = CONVENC(MSG,TRELLIS,INIT_STATE)
[CODE FINAL_STATE] = CONVENC(MSG,TRELLIS,INIT_STATE)
```

其中，MSG 为输入信息；TRELLIS 指明卷积编码所使用的结构（借助函数 POLY2TRELLIS 和 ISTRELLIS 可得到其结构）。

◆ **cyclgen 函数**

cyclgen 函数用于产生循环码的生成矩阵和校验矩阵。其调用格式为：

```
H = CYCLGEN(N, P)
H = CYCLGEN(N, P, OPT)
[H, G] = CYCLGEN(N, P, OPT)
[H, G] = CYCLGEN(N, P, OPT)
```

其中，N 为给定的码字长；P 为生成多项式；OPT 为可选参数，当 OPT = 'nonsys'时，生成一个非系统的循环校验阵，当 OPT = 'system'时，则生成一个系统的循环校验阵；H 为校验阵；G 为生成矩阵。

◆ **cyclpoly 函数**

cyclpoly 函数用于产生循环码的生成多项式。其调用格式为：

```
P - CYCLPOLY(N, K)
P = CYCLPOLY(N, K, OPT)
```

其中，N 为给定的码字长；K 为信息元长度；OPT 为可选参数。当 OPT = 'min'时，寻找最小阶数循环生成多项式；当 OPT = 'max'时，寻找最大阶数循环生成多项式；当 OPT = 'all'时，对所有给定阶数的多项式进行搜索；当 OPT = L 时，搜索有 L 项的生成多项式。

◆ **decode 函数**

decode 函数是通用的分组码译码器函数。通过该函数可以采用多种差错控制译码技

术进行译码，从而恢复出原来的信号，译码时的函数调用方式必须与编码时相同。其调用格式为：

```
MSG = ENCODE(CODE, N, K, METHOD)
MSG = DECODE(CODE, N, K, METHOD, P_POLY)
[MSG, ERR, CCODE, CERR] = DECODE(CODE, N, K, METHOD, P_POLY)
```

其中，MSG为译码后的输出；ERR为错误的码字数；CCODE为纠正的错误码字和在CODE中发现的错误字；CERR指出错误的码字是否超过了所采用的译码方式的纠错能力；N为码元长度；K为信息元长度；P_POLY是GF(2)域中的N阶多项式；METHOD是选取的译码方式，可选择的译码方式有：hamming(海明码)、linear(线性分组码)、cyclic(循环码)、bch(BCH码)和rs(R-S码)等。

◆ **encode函数**

encode函数是通用的分组码编码器函数，通过该函数可以采用多种差错控制编码技术进行编码。其调用格式为：

```
CODE = ENCODE(MSG, N, K, METHOD, OPT)
```

其中，参数METHOD的取值和OPT的含义如表5-6所示，其余参数的含义请参见decode函数。

表5-6 encode函数的参数表

METHOD	OPT的含义	说　明
'HAMMING'	可用来指定一个原始多项式	海明编码
'LINEAR'	可用来指定一个生成矩阵	线性分组码
'CYCLIC'	可用来指出循环码多项式	循环码
'BCH'	可用来指定一个BCH码的生成多项式	BCH码
'RS'	可用来指定一个R-S码的生成多项式	R-S码

◆ **gen2par函数**

gen2par函数将生成矩阵G的标准形式转换成校验矩阵，也可用于GF(2)域中，将校验矩阵转换成生成多项式。其调用格式为：

```
H = GEN2PAR(G)
```

◆ **gfweight函数**

gfweight函数用于计算线性分组码的最小码距。其调用格式为：

```
WT = GFWEIGHT(G)
WT = GFWEIGHT(G, GH_FLAG)
```

其中，G为给定的生成矩阵；GH_FLAG用于指出G的含义。当GH_FLAG = 'gen'时，G为一生成矩阵；当GH_FLAG = 'par'时，G是一校验矩阵；当GH_FLAG = n时，G是循环码或BCH码的生成多项式，n为码字长度。

◆ **hammgen函数**

hammgen函数用于产生海明码的校验矩阵和生成矩阵。其调用格式为：

```
H = HAMMGEN(M)
H = HAMMGEN(M, P)
[H, G] = HAMMGEN(M, P)
[H, G, N, K] = HAMMGEN(M, P)
```

其中,H为校验矩阵;M(M≥3)为一整数;相应的海明码的码长为 $N=2^M-1$;信息元长度为 $K=2^M-M-1$;G为生成矩阵。

◆ **rsdecof 函数**

rsdecof 函数用于对已经编码的 ASCII 文件用 R-S 码进行译码。其调用格式为:

```
RSDECOF(FILE_IN, FILE_OUT)
```

此函数格式用于将一 ASCII 文件 FILE_IN 用(127,117)R-S 码进行译码,这种 R-S 码可纠 5 位错,并将译码输出到 FILE_OUT 文件中。FILE_IN 和 FILE_OUT 都是文件指针。

```
RSDECOF(FILE_IN, FILE_OUT, ERR_COR)
```

此函数格式用于将一 ASCII 文件 FILE_IN 用(127, 127-2*ERR_COR)R-S 码进行译码,这种 R-S 码的纠错能力是 ERR_COR,并将译码输出到 FILE_OUT 文件中。FILE_IN 和 FILE_OUT 都是文件指针。

◆ **rsencof 函数**

rsencof 函数用于对 ASCII 文件进行 R-S 码编码。其调用格式为:

```
RSENCOF(FILE_IN, FILE_OUT)
```

此函数格式用于将一 ASCII 文件 FILE_IN 用(127,117)R-S 码进行编码,这种 R-S 码可纠 5 位错,并将编码输出到 FILE_OUT 文件中。FILE_IN 和 FILE_OUT 都是文件指针。

```
RSENCOF(FILE_IN, FILE_OUT, ERR_COR)
```

此函数格式用于将一 ASCII 文件 FILE_IN 用(127, 127-2*ERR_COR)R-S 码进行编码,这种 R-S 码的纠错能力是 ERR_COR,并将编码输出到 FILE_OUT 文件中。FILE_IN 和 FILE_OUT 都是文件指针。

◆ **rspoly 函数**

rspoly 函数用于产生 R-S 码的生成多项式。其调用格式为:

```
Pg = RSPOLY(N, K)
Pg = RSPOLY(N, K, M)
Pg = RSPOLY(N, K, TP)
[Pg, T] = RSPOLY(N, K, TP)
```

其中,N为 R-S 码的码长;$N=2^M-1$;$M(M\geq3)$为一整数;K为信息元长度,且K必须为一奇数,其纠错能力为$(N-K)/2$;输出 Pg 为 $GF(2^M)$域中的一个多项式;TP为N行M列的矩阵,矩阵中的所有元素均在 $GF(2^M)$域中。

TP 也可由下面语句生成:

```
TP = gftuple([-1:2^m-2]', M, 2);
```

其中,参数 T 为 R-S 的纠错能力。

◆ **syndtable 函数**

syndtable 函数用于产生综合译码表。其调用格式为:

```
T = SYNDTABLE(H)
```

其中,H为线性分组的校验矩阵。

5.3.5 纠错控制编码/解码底层函数

MATLAB通信系统工具箱提供了如表5-7所示的纠错控制编码/解码底层函数。

表5-7 纠错控制编码/解码底层函数

纠错控制编码/解码底层函数	说　明
BCHDECO	BCH码译码
BCHENCO	BCH码编码
RSDECO	R-S码译码
RSDECODE	用指数形式进行R-S译码
RSENCO	R-S码编码
RSENCODE	用指数形式进行R-S编码

表中各函数的使用方法及功能如下:

◆ **bchdeco函数**

bchdeco函数用于BCH码译码,其调用格式为:

```
MSG = BCHDECO(CODE, K, T)
MSG = BCHDECO(CODE, K, T, ORD)
[MSG, ERR] = BCHDECO(CODE, K, T, ORD)
[MSG, ERR, CCODE] = BCHDECO(CODE, K, T, ORD)
```

其中,MSG为恢复的信息码字;CODE为待译码的码字;K为信息元长度;T为纠错能力;ORD为指定的初始多项式;ERR为纠错位数;CCODE为纠正的码字。

◆ **bchenco函数**

bchenco函数用于BCH码编码。其调用格式为:

```
CODE = BCHENCO(MSG, N, K)
CODE = BCHENCO(MSG, N, K, PG)
```

其中,N为码字长度;K为信息元长;PG为生成多项式。

◆ **rsdeco函数**

rsdeco函数用于R-S码译码。其调用格式为:

```
MSG = RSDECO(CODE, N, K)
MSG = RSDECO(CODE, N, K, TYPE_FLAG)
[MSG, ERR] = RSDECO(CODE, N, K, TYPE_FLAG)
[MSG, ERR, CCODE] = RSDECO(CODE, N, K, TYPE_FLAG)
```

其中,N为R-S码的码长,$N=2^M-1$;$M(M\geq3)$为一整数;K为信息元长度;ERR为纠错位数;CCODE为纠正的码字;ERR_C表示在相应的CCODE的列的错误数;TYPE_

FLAG 指定 CODE 的输入格式。TYPE _ FLAG 默认值为'binary',表示 CODE 为二进制码;当 TYPE _ FLAG = 'decimal'时,表示 CODE 为[0,N]的整数;当 TYPE _ FLAG = 'power'时,表示 CODE 为 GF(2^M)域的指数。

◆ **rsdecode 函数**

rsdecode 函数用指数形式进行 R-S 译码。其调用格式为:

```
MSG = RSDECODE(CODE, K)
MSG = RSDECODE(CODE, K, TP)
[MSG, ERR] = RSDECODE(CODE, K, TP)
[MSG, ERR, CCODE] = RSDECODE(CODE, K, TP)
```

其中,CODE 为待恢复的信息的码字,它为一 N 行的矩阵,$N=2^M-1$;$M(M\geqslant3)$为一整数;K 为信息元长度,通常为奇数,并且 R-S 码的纠错能力为 T = floor((N-K) /2);恢复的信息存储在 MSG 中;TP 为 N 行 M 列的矩阵,矩阵中的所有元素均在 GF(2^M)域中;CCODE 为已经纠正的码字;ERR 给出纠错的信息。如果 ERR 为非负的整数,则表示已经纠正的错误数;如果 ERR 为负数,则表示错误数超过了纠错能力。

TP 也可由下面语句生成:

```
TP = gftuple([-1:2^m-2]', M, 2);
```

◆ **rsenco 函数**

rsenco 函数采用 R-S 码进行编码。其调用格式为:

```
CODE = RSENCO(MSG, N, K)
CODE = RSENCO(MSG, N, K, TYPE_FLAG)
CODE = RSENCO(MSG, N, K, TYPE_FLAG, PG)
[CODE, ADDED] = RSENCO(MSG, N, K, TYPE_FLAG, PG)
```

其中,N 为 R-S 码的码长,$N=2^M-1$;$M(M\geqslant3)$为一整数;K 为信息元长度,K<N;PG 为 R-S 码的生成多项式;ADDED 表示在 MSG 中加入了的列数;TYPE _ FLAG 指定 CODE 的输入格式。TYPE _ FLAG 默认值为'binary',表示 CODE 为二进制码;当 TYPE _ FLAG = 'decimal'时,表示 CODE 为[0,N]的整数;当 TYPE _ FLAG = 'power'时,表示 CODE 为 GF(2^M)域的指数。

◆ **rsencode 函数**

rsencode 函数用指数形式进行 R-S 编码。其调用格式为:

```
CODE = RSENCODE(MSG, PG, N)
CODE = RSENCODE(MSG, PG, N, TP)
```

其中,MSG 为要编码的信息;PG 为生成多项式;N 为 R-S 码的码长,$N=2^M$;$M(M\geqslant3)$为一整数;TP 列出 GF(2^M)域中所有元素。

5.3.6 调制/解调函数

MATLAB 通信系统工具箱提供了如表 5-8 所示的调制/解调函数。

表 5-8　调制/解调函数表

调制/解调函数	说　明
ADEMOD	模拟通带解调
ADEMODCE	模拟基带解调
AMOD	模拟通带调制
AMODCE	模拟基带调制
APKCONST	绘出 ASK-PSK 调制的环形星座图
DDEMOD	数字通带解调
DDEMODCE	数字基带解调
DEMODMAP	模拟信号到数字信号的映射
DMOD	数字通带调制
DMODCE	数字基带调制
MODMAP	数字信号到模拟信号的映射
QASKDECO	矩形星座 QASK 的逆映射
QASKENCO	绘制 QASK 矩形星座图

表中各函数的使用方法及功能如下：

◆ **ademod 函数**

该函数可以采用多种方法对已调制的模拟信号进行解调。其调用格式为：

```
Z = ADEMOD(Y, Fc, Fs, METHOD...)
```

其中，Y 为已调制信号；Fc(Hz)为载波频率；Fs(Hz)为采样频率，Fc＞Fs；METHOD 代表所采用的解调方法，其取值及含义如表 5-9 所示。

表 5-9　METHOD 的选取及含义

METHOD	说　明
AMDSB-TC	双边带载波幅度解调
AMDSB-SC	双边带抑制载波幅度解调
AMSSB	单边带抑制载波幅度解调
QAM	正交幅度解调(QAM)
FM	频率解调
PM	相位解调

◆ **ademodce 函数**

ademodce 函数可以采用多种方法对已调制的模拟信号进行复包络方式解调。其调用格式为：

```
Z = ADEMODCE(Y, Fs, METHOD...)
```

其中，Y 为已调制的复包络信号；Fs(Hz)为采样频率；METHOD 代表所采用的解调

方法,其取值及含义如表 5-9 所示。

◆ **amod 函数**

amod 函数可以采用多种方法对输入的模拟信号进行调制。其调用格式为:

```
Y = AMOD(X, Fc, Fs, METHOD...)
```

其中,X 为消息信号;Fc(Hz)为载波频率;Fs(Hz)为采样频率,Fc>Fs;METHOD 代表所采用的调制方法,其取值及含义如表 5-9 所示。

◆ **amodce 函数**

amodce 函数可以采用多种方法对输入的模拟信号进行调制。其调用格式为:

```
Y = AMODCE(X, Fs, METHOD...)
```

其中,X 为消息信号;Fs(Hz)为采样频率;METHOD 代表所采用的调制方法,其取值及含义如表 5-9 所示。

◆ **apkconst 函数**

apkconst 函数用于绘制 ASK-PSK 调制的环形星座图。其调用格式为:

```
APKCONST(NUMSIG, AMP, PHASE)
APKCONST(NUMSIG, AMP)
APKCONST(NUMSIG)
Y = APKCONST(NUMSIG, AMP, PHASE)
Y = APKCONST(NUMSIG, AMP)
Y = APKCONST(NUMSIG)
```

其中,NUMSIG 为星座的点数;AMP 为圆的半径;PHASE 为相位。前三种函数格式将绘出星座图;后三种函数格式不绘出图形,只返回一复数向量,其实部为同相分量,虚部为正交分量。

◆ **ddemod 函数**

ddemod 函数用于通带信号的数字解调。其调用格式为:

```
Z = DDEMOD(Y, Fc, Fd, Fs, METHOD...)
```

其中,Y 为数字调制信号;Fc(Hz)为载波频率;Fd(Hz)为采样频率;Fs(Hz)为计算出的采样频率,比值 Fs/Fd 必须为大于 0 的整数;METHOD 代表所采用的解调方法,其取值及含义如表 5-10 所示。

表 5-10 METHOD 的取值及含义

METHOD	说　明
ASK	M 进制幅度键控
PSK	M 进制相移键控
QASK	正交幅度键控
FSK	M 进制频移键控
MSK	最小频移键控
SAMPLE	转换采样频率

◆ **ddemodce 函数**

ddemodce 函数用于基带信号的数字解调。其调用格式为:

Z = DDEMODCE(Y, Fd, Fs, METHOD...)

其中,Y 为数字调制信号;Fd(Hz)为采样频率;Fs(Hz)为计算出的采样频率,比值 Fs/Fd 必须为一正整数;METHOD 代表所采用的解调方法,其取值及含义如表 5-10 所示。

◆ **demodmap 函数**

demodmap 函数是 MODMAP 的逆映射函数,它将模拟信号 Y 映射为数字信号 Z。其调用格式为:

Z = DEMODMAP(Y, Fd, Fs, METHOD...)

METHOD 代表所采用的解调方法,其取值及含义如表 5-10 所示。

◆ **modmap 函数**

modmap 函数将数字信号 X 映射为模拟信号 Y。其调用格式为:

MODMAP(METHOD...)

此函数格式可以绘制出指定映射方法的信号星座图。参数 METHOD 的取值及含义如表 5-11 所示。

Y = MODMAP(X, Fd, Fs, METHOD...)

此函数格式将数字信号 X 映射为模拟信号 Y,只进行映射,不进行调制。参数 METHOD 的取值及含义如表 5-11 所示。

表 5-11 METHOD 的取值及含义

METHOD	说　明
ASK	M 进制幅度键控
PSK	M 进制相移键控
QASK	正交幅度键控
FSK	M 进制频移键控
MSK	最小频移键控

◆ **dmod 函数**

dmod 函数用于通带信号的数字调制。其调用格式为:

Y = DMOD(X, Fc, Fd, Fs, METHOD...)

其中,X 为消息信号;Fc(Hz)为载波频率;Fd(Hz)为符号频率;Fs(Hz)为 Y 的采样频率且要满足条件 Fs>Fc,比值 Fs/Fd 为大于 0 的正整数;参数 METHOD 的取值及含义如表 5-11 所示。

◆ **dmodce 函数**

dmodce 函数用于基带信号的数字调制,它的输出为数字调制信号的复包络。其调用格式为:

Y = DMODCE(X, Fd, Fs, METHOD...)

其中,Fd(Hz)是消息信号 X 的采样频率;Fs(Hz)为输出 Y 的采样频率,且要满足条件 Fs/Fd 为大于 0 的正整数;METHOD 的取值及含义如表 5-11 所示。

◆ **qaskdeco 函数**

qaskdeco 函数根据 QASK 矩形星座图,把信号 X、Y 中的数值对应的信息映射出来。

其调用格式为：

MSG=QASKDECO(X, Y, M)

MSG=QASKDECO(X, Y, M, MINMAX)

其中，X中存放I路数值，即同相分量；Y中存放Q路数值，即正交分量；M表示几进制，且M必须为2的整数次幂；MINMAX给出X和Y的最大最小值，其形式为：

$$MINMAX=\begin{vmatrix} X_minX_max \\ Y_minY_max \end{vmatrix} \tag{5-1}$$

◆ **qaskenco 函数**

qaskenco 函数用于绘制QASK矩形星座图。其调用格式为：

QASKENCO(M)

此函数格式用于绘出M进制的QASK矩形星座图，M必须为2的整数次幂。

QASKENCO(MSG, M)

此函数格式用于绘出数字信号MSG在星座图上的位置，MSG中的元素为[0,M-1]内的整数。

[X, Y] = QASKENCO(M)

此函数格式输出QASK矩形星座的同相分量X和正交分量Y。

[X, Y] = QASKENCO(MSG, M)

此函数格式将信息信号MSG在M进制矩形星座图中的I、Q路信息输出到变量X和Y中。

5.3.7 特殊滤波器函数

MATLAB通信系统工具箱提供了如表5-12所示的特殊滤波器函数。

表5-12　特殊滤波器函数表

特殊滤波器函数	说　明
HANK2SYS	将HANKEL矩阵转换成线性系统模型
HILBIIR	希尔伯特(HILBERT)IIR滤波器设计
RCOSFLT	对输入信号进行升余弦滤波
RCOSINE	升余弦滤波器设计

表中各函数的使用方法及功能如下：

◆ **hank2sys 函数**

hank2sys 函数用于将Hankel矩阵转换为线性系统模型。其调用格式为：

[NUM, DEN] = HANK2SYS(H, INI, TOL)

[NUM, DEN, SV] = HANK2SYS(H, INI, TOL)

这两个函数调用格式用于将Hankel矩阵H转换为线性系统的传递函数模型，NUM和DEN分别为线性系统传递函数的分子和分母；INI为该系统在0时刻的冲击响应。当TOL>1时，TOL表示转换的阶数；当TOL<1时，TOL表示基于单模值的阶数选择的容

差,默认值为 0.01。如果函数的转换采用的是单模值分解法,SVD 即为单模值。

[A, B, C, D] = HANK2SYS(H, INI, TOL)

[A, B, C, D, SVD] = HANK2SYS(H, INI, TOL)

这两个函数格式用于将 Hankel 矩阵 H 转换为线性系统的状态空间模型,A、B、C 和 D 为线性系统的状态空间模型矩阵,SVD 为单模值。

◆ **hilbiir 函数**

hilbiir 函数将用 IIR 滤波器实现 Hilbert 变换。其调用格式为:

HILBIIR;

此函数格式用于绘出理想 Hilbert 变换滤波器和一个设计的 4 阶 Hilbert 数字滤波器的冲击响应图,滤波器的采样时间为 2/7 秒,时延为 1 秒。

HILBIIR(TS)

此函数格式用于绘出理论上的理想 Hilbert 变换和一个设计的 4 阶 Hilbert 数字滤波器的冲击响应图,滤波器的采样时间为 TS 秒,群延时为 3.5TS 秒,滤波器的阶数用容差的默认值 TOL=0.05 来确定。

HILBIIR(TS, DLY)

此函数格式用于绘出所设计的采样时间为 TS 秒、群延时为 DLY 的 Hilbert 数字滤波器的冲击响应图。

HILBIIR(TS, DLY, BANDWIDTH)

此函数格式用于绘出所设计的采样时间为 TS 秒、群延时为 DLY 的 Hilbert 数字滤波器的冲击响应图,滤波器设计时采用了输入信号压缩方式,输入信号的带宽为 BANDWIDTH,当 BANDWIDTH =0 或者 BANDWIDTH>1/(TS/2)时,没有采用压缩。

HILBIIR(TS, DLY, BANDWIDTH, TOL)

此函数格式用于绘出所设计的采样时间为 TS 秒、群延时为 DLY 的 Hilbert 数字滤波器的冲击响应图。输入信号的带宽为 BANDWIDTH;容差为 TOL;滤波器的阶数用"截项单模值/最大单模值"<TOL 来确定。当 TOL≥1 时,TOL 即为滤波器的阶数,当 BANDWIDTH=0 或者 BANDWIDTH>1/(TS/2)时,滤波器的阶数恰好为 TOL,此时没有采用压缩;否则,滤波器的阶数为 TOL+max(3,ceil(TOL)/2),且采用了压缩方式。

[NUM, DEN] = HILBIIR(...)

[NUM, DEN, SV] = HILBIIR(...)

[A, B, C, D] = HILBIIR(...)

[A, B, C, D, SV] = HILBIIR(...)

在这四种调用方式中,NUM 和 DEN 分别为滤波器传递函数的分子和分母;A、B、C 和 D 为滤波器状态空间模型矩阵;SV 为单模值。

◆ **rcosflt 函数**

rcosflt 函数采用升余弦滤波器对输入信号进行滤波。其调用格式为:

Y = RCOSFLT(X, Fd, Fs, TYPE_FLAG, R, DELAY)

Y = RCOSFLT(X, Fd, Fs, TYPE_FLAG, R)

Y = RCOSFLT(X, Fd, Fs, TYPE_FLAG)

Y = RCOSFLT(X, Fd, Fs)

Y = RCOSFLT(X, Fd, Fs, TYPE_FLAG, R, DELAY, TOL)

```
[Y, T] = RCOSFLT(...)
```

其中,Fd(Hz)为输入信号 X 的采样频率;Fs(Hz)为输出 Y 的采样频率,且 Fs 必须为 Fd 的整数倍;TYPE_FLAG 为滤波器类型;R 为滚降系数;DELAY 为群延时;TOL 为 IIR 滤波器的容差;T 为时间向量。TYPE_FLAG 的取值及含义如表 5-13 所示。

表 5-13 TYPE_FLAG 的取值及含义

TYPE_FLAG	说 明
'FIR'	FIR 滤波器
'IIR'	IIR 滤波器设计
'NORMAL'	一般的升余弦滤波器
'SQRT'	平方根升余弦滤波器
'FS'	把输入信号采样频率转换到 FS
'FILTER'	滤波器类型由用户指定

◆ **rcosine 函数**

rcosine 函数用于设计升余弦滤波器。其调用格式为:

```
NUM = RCOSINE(Fd, Fs)
```

此函数格式用于设计 FIR 升余弦滤波器。Fd 为数字转换频率;Fs 为滤波器采样频率,且比值 Fs/Fd 必须为一正整数;滚降系数为 0.5;时延为 3/Fd 秒。

```
[NUM, DEN] = RCOSINE(Fd, Fs, TYPE_FLAG)
[NUM, DEN] = RCOSINE(Fd, Fs, TYPE_FLAG, R)
[NUM, DEN] = RCOSINE(Fd, Fs, TYPE_FLAG, R, DELAY)
[NUM, DEN] = RCOSINE(Fd, Fs, TYPE_FLAG, R, DELAY, TOL)
```

其中,TYPE_FLAG 的取值及含义如表 5-14 所示。

表 5-14 TYPE_FLAG 取值及含义

TYPE_FLAG	说 明
'FIR'	FIR 升余弦滤波器
'IIR'	IIR 升余弦滤波器
'NORMAL'	一般的升余弦滤波器
'SQRT'	平方根升余弦滤波器
'DEFAULT'	默认值'FIR'或'NORMAL'

5.3.8 特殊滤波器底层函数

MATLAB 通信系统工具箱提供了如表 5-15 所示的特殊滤波器底层函数。

表 5－15　特殊滤波器底层函数表

特殊滤波器底层函数	说　　明
RCOSFIR	升余弦 FIR 滤波器设计
RCOSIIR	升余弦 IIR 滤波器设计

表中各函数的使用方法及功能如下：

◆ **rcosfir 函数**

rcosfir 函数用于设计升余弦 FIR 滤波器。其调用格式为：

B=RCOSFIR(R, N_T, RATE, T)

其中，T(秒)为输入信号的采样周期；RATE 为过采样率；R 为滚降系数；N_T 为长度为 2 的标量或向量。当 N_T 为标量时，滤波器的采样点数为 2×N_T+1；当 N_T 为向量时，滤波器的采样点数为 N_T(2)－N_T(1)+1。

B=RCOSFIR(R, N_T, RATE, T, FILTER_TYPE)

其中，当 FILTER_TYPE='sqrt'时，设计平方根升余弦滤波器；当 FILTER_TYPE='normal'时，设计一般的升余弦滤波器，它为默认值。

RCOSFIR(R, N_T, RATE, T, FILTER_TYPE, COL)

[B, Sample_Time] = RCOSFIR(...)

其中，COL 为绘图的颜色，Sample_Time 为采样时间。

◆ **rcosiir 函数**

rcosiir 函数用于设计升余弦 IIR 滤波器。其调用格式为：

[NUM, DEN] = RCOSIIR(R, T_DELAY, RATE, T, TOL)

该函数格式用给定的滚降系数 R 设计一 IIR 滤波器。此 IIR 滤波器与 FIR 升余弦滤波器近似，T_DELAY 是给定的时延，其值等于采样周期 T 的整数倍；RATE 为每一采样间隔内的采样点数；滤波器的阶数由容差 TOL 来确定。

[NUM, DEN] = RCOSIIR(R, T_DELAY, RATE, T, TOL, FILTER_TYPE)

该函数格式用于设计一 IIR 滤波器，此 IIR 滤波器与 FIR 平方根升余弦滤波器近似，FILTER_TYPE =='sqrt'。

RCOSIIR(...)

RCOSIIR(..., COLOR)

这两个函数格式用于绘出升余弦滤波器的时域和频率域响应，COLOR 为指定的绘图颜色。

[NUM, DEN, SAMPLE_TIME] = RCOSIIR(...)

该函数格式调用后返回 IIR 滤波器的传递函数和采样时间。

5.3.9　实用工具函数

MATLAB 通信系统工具箱提供了如表 5－16 所示的实用工具函数。

表 5-16　实用工具函数表

实用工具函数	说　明
BI2DE	二进制向量转换为十进制向量
DE2BI	十进制向量转换为二进制向量
ERF	误差函数
ERFC	互补误差函数
ISTRELLIS	格型结构的有效性检验
MARCUMQ	广义 Q 函数
OCT2DEC	将八进制数转换为十进制数
POLY2TRELLIS	将卷积码多项式转换为格型描述
VEC2MAT	将向量转换为矩阵

表中各函数的使用方法及功能如下：

◆ **bi2de 函数**

bi2de 函数用于实现二进制向量向十进制向量的转换。其调用格式为：

D=BI2DE(B)

其中，B 为二进制向量；D 为转换后的十进制向量。B 也可为矩阵，此时该函数按 B 的行进行转换，得到的 D 为一列向量。

◆ **de2bi 函数**

de2bi 函数用于实现十进制向量向二进制向量的转换。其调用格式为：

B = DE2BI(D)

B = DE2BI(D,N)

其中，D 为非负十进制整数向量；B 为转换后的二进制矩阵；N 指定输出的二进制位数。

◆ **erf 函数**

erf 函数用于计算函数的误差，误差函数的定义为：

$$erf(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2/2}dt \tag{5-2}$$

函数调用格式为：

Y = ERF(X)

◆ **erfc 函数**

互补误差函数的定义为：

$$erfc(x) = 1 - erf(x) = \frac{2}{\sqrt{\pi}}\int_x^{\infty} e^{-t^2/2}dt \tag{5-3}$$

函数调用格式为：

Y = ERFC(X)

◆ **istrellis 函数**

istrellis 函数用于检验输入是否为有效的格型结构。其调用格式为：

```
[ISOK, STATUS] = ISTRELLIS(S)
```

若 S 为一有效的格型结构，则函数调用后 ISOK 为 1，这时 STATUS 为空字符串，否则 ISOK 为 0；STATUS 为一字符串。

◆ **marcumq 函数**

marcumq 函数实现广义 Q 函数的计算。其调用格式为：

```
MARCUMQ(A,B,M)
```

其中，A、B 均为实数，且为非负；M 必须为整数。A、B 和 M 均为广义 Q 函数中的参数。

◆ **oct2dec 函数**

该函数用于实现八进制数到十进制数的转换，函数的调用格式为：

```
D = BI2DE(O)
```

其中，O 为八进制数，也可以为八进制数向量。

◆ **poly2trellis 函数**

该函数实现卷积码多项式到格型结构的转换。其调用格式为：

```
TRELLIS = POLY2TRELLIS(CONSTRAINLENGTH, CODEGENERATOR)
```

其中，CODEGENERATOR 表示卷积码多项式；CONSTRAINLENGTH 表示长度。

◆ **vec2mat 函数**

vec2mat 函数用于实现向量到矩阵的转换。其调用格式为：

```
MAT = VEC2MAT(VEC, MATCOL)
```

其中，VEC 为输入向量；MATCOL 为转换后的矩阵的列数。

5.3.10 伽罗华域计算函数

MATLAB 通信系统工具箱提供了如表 5-17 所示的伽罗华域(Galois Field)计算函数。

表 5-17　伽罗华域计算函数表

伽罗华域计算函数	说　明
GFADD	伽罗华域中的多项式加法计算
GFCONV	伽罗华域中的多项式乘法计算
GFCOSETS	伽罗华域中的割圆陪集计算
GFDECONV	伽罗华域中的逆卷积计算
GFDIV	伽罗华域中的除法计算
GFFILTER	基本伽罗华域的多项式数据过滤计算
GFLINEQ	伽罗华域中的方程 $AX=B$ 求解
GFMINPOL	寻找伽罗华域中的最小多项式
GFMUL	将向量转换为矩阵
GFPLUS	伽罗华域中的加法计算
GFPRETTY	伽罗华域中多项式表示

（续表）

伽罗华域计算函数	说　明
GFPRIMCK	伽罗华域中多项式的可约性检测
GFPRIMDF	输出指定维数的伽罗华域中的原始多项式
GFPRIMFD	寻找伽罗华域中的原始多项式
GFRANK	计算伽罗华域中矩阵的秩
GFREPCON	伽罗华域中多项式表示方式的转换
GFROOTS	求解伽罗华域中多项式的根
GFSUB	伽罗华域中多项式的减法计算
GFTRUNC	伽罗华域中多项式的最短化处理
GFTUPLE	伽罗华域中多项式的约简和转化

5.4 通信系统仿真实例

利用 MATLAB 通信系统工具箱进行通信仿真，方法有两种：一种是利用工具箱中 MATLAB 通信函数进行计算方式的数据流仿真；另一种是利用工具箱中 SIMULINK 通信仿真模型库进行动态方式的时间流仿真。换句话说，在 SIMULINK 仿真中，每一时刻，所有的功能模型均同时在执行，而在 MATLAB 仿真中，功能函数是随数据流依次执行的，即数据流处理是一级一级传递的。因此，在绝大多数情况下，通信系统仿真均利用 SIMULINK 环境来进行，只有在特殊情况下，如基于仿真的优化设计，才调用 MATLAB 函数计算仿真。

下面通过实例说明这两种仿真方法的运用，以及如何使用通信系统工具箱中的仿真模块和仿真命令来对通信系统进行仿真。

例 5－2 创建一个信道噪声模型。其中，信源采用伯努利随机二进制产生器，信道采用二进制对称信道，并检测和输出由噪声导致的错误率。

分析：要建立此仿真模型，需要下列仿真模块：

- Bernoulli RandomBinary Generator（伯努利随机二进制产生器），从 Comm Sources（信源）子模块集中得到。
- Binary Symmetric Channel（二进制对称信道），从 Channels（信道）子模块集中可以得到。
- Error Rate Calculation（错误率计算）模块，从 Comm Sinks（信号接收）子模块集中得到。
- Display（显示器）模块，此模块可以从 Simulink（基本子模块集）下的 Sinks（信号接收）子模块集中得到。

利用这些仿真模块，可建立如图 5－31 所示的信道噪声仿真模型。运行该仿真模型一段时间后，将得到如图 5－32 所示的结果。在 Display 模块中显示的结果中，SER（Symbol Error Rate）为当前的符号错误率，在仿真过程中是不断变化的；TE（Total Errors）为接收端接收到的总错误数；TS（Total Symbols）为信道传输的总符号数。图 5－32 中显示的

是仿真时间为 5000 秒的结果：SER = 0.01028，TE = 514，TS = 50000。可见，这样的传输过程的出错率是非常大的，需要采取措施来减少差错率。

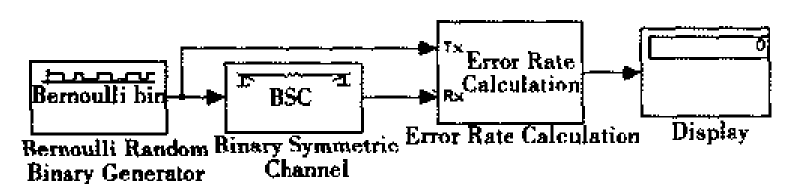


图 5－31　例 5－2 的信道噪声仿真模型

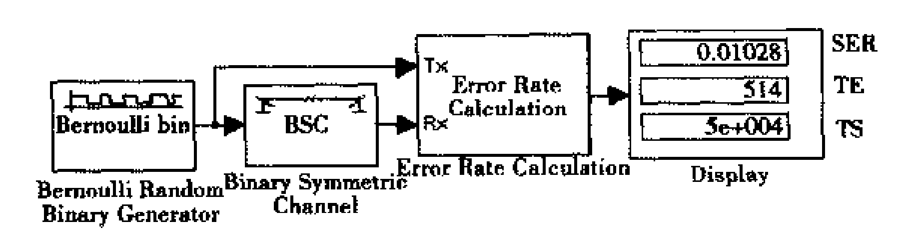


图 5－32　例 5－2 的运行仿真结果

例 5－3　为了减少例 5－2 中的信号传输过程中的出错率，在例 5－2 中的信道噪声模型中加入信道编码技术，并对比其结果。

在本例中，选取 Hamming Code(海明码)进行信道编码，由于编码和译码总是成对使用，故需要同时加入编码和译码仿真模块。从 Channel Codes(信道编码)子模块集中将海明编码和译码模块加入到图 5－31 所示的图中，将得到图 5－33 所示的仿真模型。

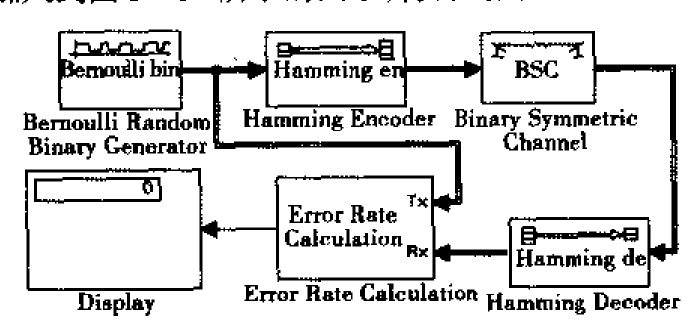


图 5－33　例 5－3 中的仿真模型

为了与例 5－2 中的仿真结果进行对比，故仿真时间仍然确定为 5000 秒，仿真结果如图 5－34 所示，SER = 0.00122，TE = 61，TS = 50000。由此可见，在总的传输符号数不变的情况下，采用信道编码技术后，传输的错误率已大大减小，SER 仅为先前的 11.87%，TE 仅为先前的 11.87%。

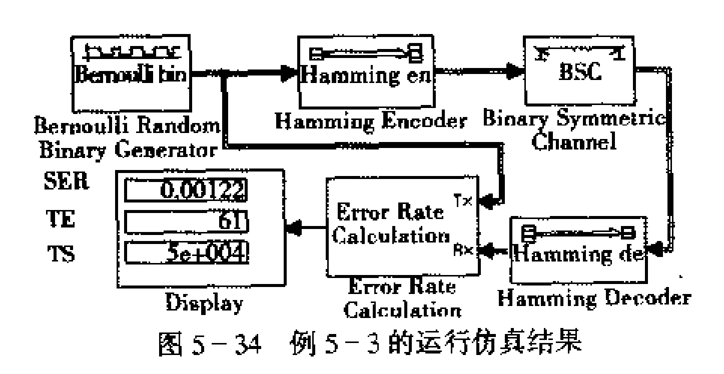


图 5－34　例 5－3 的运行仿真结果

例 5-4 建立一数字调制信道模型，噪声为加性高斯白噪声。

本例的调制方式采用 BPSK(二进制相移键控)，BPSK 调制器和解调器仿真模块在 Modulation(调制)子模块集下的数字基带调制子模块集中的 PM 子模块组中，AWGN 模块在 Channel(信道)子模块集中。建立的仿真模型如图 5-35 所示。

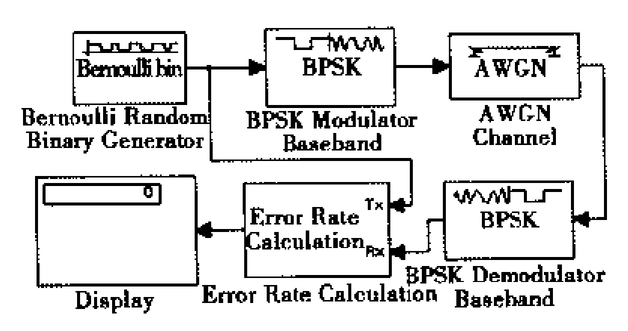


图 5-35 例 5-4 中的仿真模型

其中，BPSK 调制器和解调器的参数使用默认值，仿真时间设置为 5000 秒，AWGN 中的信噪比设置为 10dB。运行仿真模型后将得到如图 5-36 所示的结果。

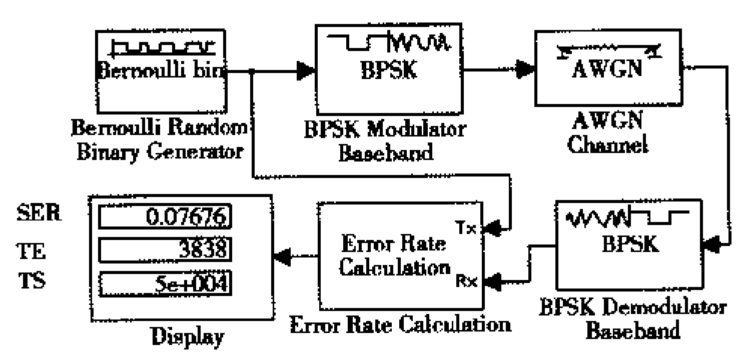


图 5-36 例 5-4 中的仿真模型

例 5-5 Frequency-shift keying(频移键控，FSK)是一种标准的调制技术，它将数字信号加载到不同频率的正弦载波上。美国的贝尔电话系统就曾采用这种调制方式。本例介绍如何建立一个用于带通信号的频移键控的仿真模型。

建立频移键控的仿真模型需要以下仿真模块：

- M-FSK Modulator Passband(M-FSK 通带调制器)模块，它位于调制子模块集下的数字调制子模块集中的 FM(频率调制)子模块组中。
- M-FSK Demodulator Passband(M-FSK 通带解调器)模块，它位于调制子模块集下的数字调制子模块集中的 FM(频率调制)子模块组中。
- Relational Operator(关系运算)模块，该模块可以从 Simulink(基本仿真子模块集)下的 Math(数学运算)子模块集中得到，用于比较信号源发出的信号与经过调制、信道传输、再解调后的信号的异同。
- Spectrum Scope(频谱示波器)模块，它位于 DSP 子模块集下的信号接收(Sinks)子模块集中，用此模块来进行信号的 FFT 计算和显示频谱。
- Scope(示波器)模块，该模块存放在 Simulink(基本仿真子模块集)下的 Sinks(信号接收)子模块集中，用于观察原信号、接收信号和关系运算模块的输出信号。

- Integer Delay(整数延迟)模块,该模块位于DSP子模块集下的Signal Operations(信号操作)子模块集中,它将信号源发出的信号进行延迟,再与接收信号作比较。

另外,还需要Bernoulli Random Binary Generator(伯努利二进制随机信号产生器)、AWGN Channel(加性高斯白噪声信道)、Error Rate Calculation(错误率计算)模块和Display(显示器)模块。整个系统的仿真模型如图5-37所示。

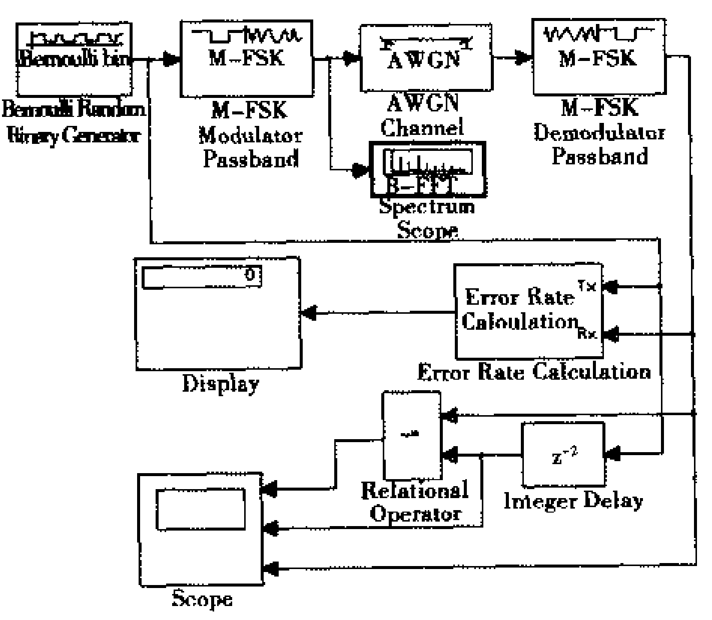


图5-37 例5.5的系统仿真模型

仿真模型中各模块的主要参数设置情况如下:

- Bernoulli Random Binary Generator:Sample time(采样时间)为1/1200。
- AWGN Channel:Es/No(信噪声比)为10,Symbol periods(符号周期)为1/1200。
- M-FSK Modulator Passband和M-FSK Demodulator Passband:采用的M-ary number(进制数)为2,Frequency separation(采样频率)为1000Hz,Symbol periods(符号周期)为1/1200,符号的Baseband samples per symbols(基带采样点数)为8,Carrier frequency(载波频率)为10000Hz,Output sample time(输出采样时间)为1/48000。
- Spectrum Scope:Buffer size(缓冲区大小)为1024,Buffer overlap(缓冲区重叠)为256,Number of spectral averages(平均谱密度)为20。

运行仿真模型后,得到Spectrum Scope(频谱示波器)的显示如图5-38所示,Scope(示波器)的显示如图5-39所示,Display(显示器)的显示如图5-40所示。

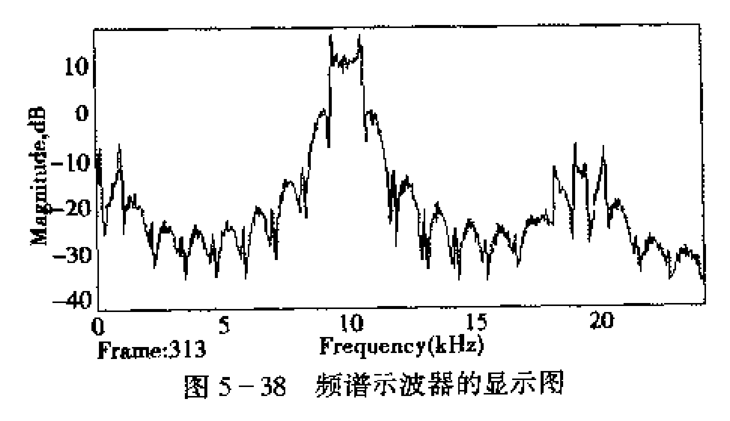


图5-38 频谱示波器的显示图

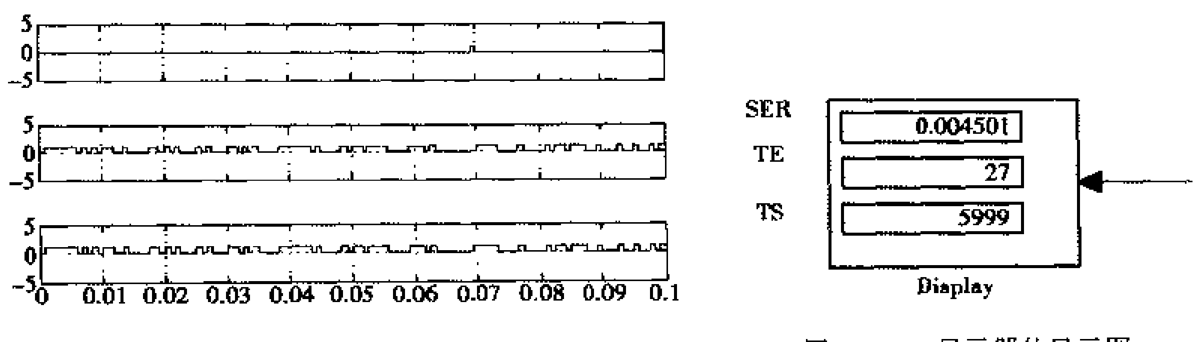


图 5－39　示波器的显示图　　　　图 5－40　显示器的显示图

提示：为了看得清楚，在图 5－39 中只给出了仿真时间为 0.1 秒的三个信号的波形，上面一个波形为接收信号与经延迟后的源信号的比较结果，中间一个波形为经延迟后的源信号的波形，下面一个为接收到的信号波形。

例 5－6　建立一个卷积码编码的通信系统仿真模型。

建立卷积码编码的通信系统仿真模型，需要以下仿真模块：

- Convolutional Encoder（卷积码编码器）：此模块存放在信道编码子模块集下的卷积码子模块集中。用于将伯努利二进制随机信号进行卷积编码，然后送入传输信道中。
- Complex to Real-Image（复数到实数的转化）模块：此模块位于 Simulink Math 子模块集中。它用于将接收到的复数信号的实部分离出来。
- Viterbi Decoder（Viterbi 译码器）：此模块存放在信道编码子模块集下的卷积码子模块集中。它用于恢复采用 Viterbi 算法进行卷积编码的信号。
- M-FSK Modulator Passband（M-FSK 通带调制器）模块：此模块位于调制子模块集下的数字调制子模块集中的 FM（频率调制）子模块组中。

另外，还需要 Bernoulli Random Binary Generator（伯努利二进制随机信号产生器）、AWGN Channel（加性高斯白噪声信道）、Error Rate Calculation（错误率计算）模块和 Display（显示器）模块等。

卷积码编码的通信系统仿真模型如图 5－41 所示。

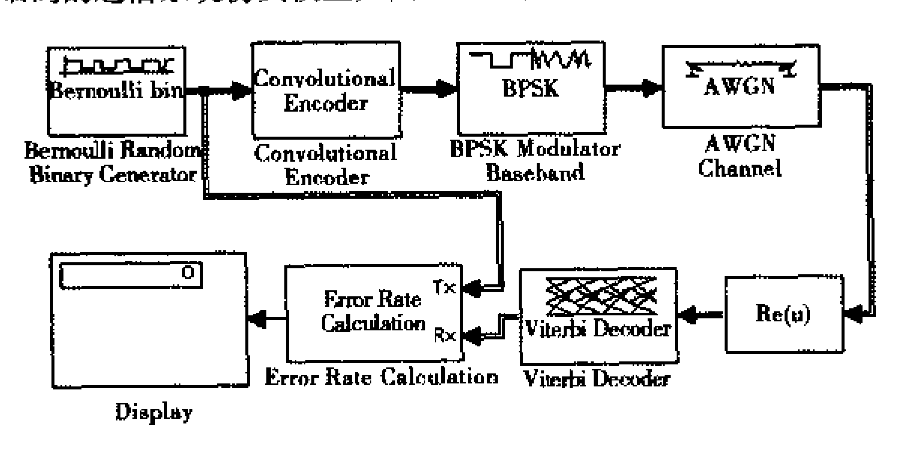


图 5－41　例 5－6 的系统仿真模型

运行结果如图 5－42 所示。

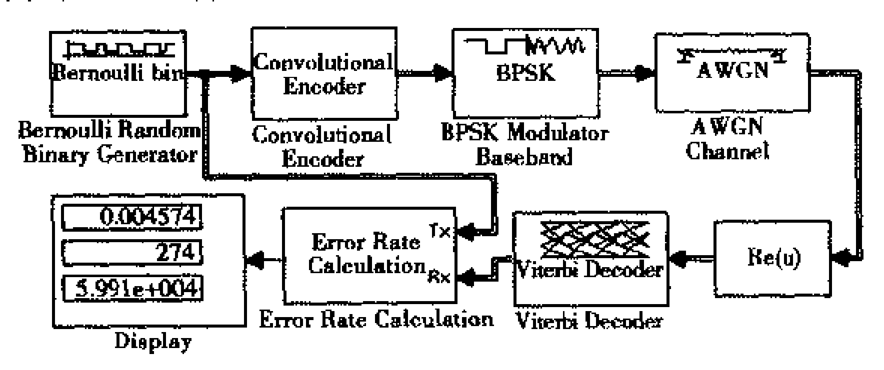


图 5－42　例 5－6 的仿真结果

例 5－7　建立一个带 BCH 纠错控制编码的 16 进制正交幅度键控调制(16-QASK)通信系统。

利用 MATLAB 的 M 文件实现通信系统仿真过程，实质上是数据流的计算处理过程。首先利用通信工具箱中的信号源产生器产生一个符合要求的信源信号；然后利用 BCH 编码函数对信源信号进行 BCH 编码；接着再用调制解调函数对编码信号进行 16-QASK 数字调制处理；最后用一个误码率计算函数将恢复信号与信源信号作出比较计算。这里，传输信道采用的是通带加性高斯白噪声信道。在仿真中，整个输入数据矢量是作为一个整体来处理的，即首先对信源矢量信号进行 BCH 编码，全部编码完成后再进行调制。一般来说，在用 MATLAB 的 M 文件进行仿真时，要查看仿真输出结果或将仿真输出结果与前面的中间结果进行比较，必须等到整个仿真计算全部完成。同时，在仿真过程中也不能根据结果来调整各项参数，也就是说，用 MATLAB 的 M 文件计算的仿真是一种静态仿真。

上述系统的 MATLAB 的 M 文件仿真程序如下：

```
%MATLAB program 5－7
%16－QASK
clear all;
clc;
M =16; N =15; K =11 ;                              % 定义进制、码长和信息长度
Fc =10000; Fd =1000; Fs =100000;                    % 定义载波频率、数字传输频率
                                                    % 和模拟频率
msg=randint(K*100,1);                               % 产生 1100×1 包含待传输信息
                                                    % 的矩阵;
                                                    % randint 产生随机整数矩阵
code=encode(msg, N, K, 'bch');                      % 使用 BCH 编码:长为 N;信
                                                    % 息长度为 K
modu=dmod(code, Fc, Fd, Fs, 'qask', M);             % 使用 16-QASK 数字调制方法;
                                                    % 进制为 M=16
std_value=0.1;                                      % 假定加性白高斯噪声的 STD
                                                    % 为0.1
modu_noise =modu+randn(length(modu), 1)*std_value;
demo =ddemod(modu_noise, Fc, Fd, Fs, 'qask', M);    % 解调过程
```

```
msg r - decode(demo, N, K, 'bch');                    % BCH解码,码长为N,信息长
                                                      % 度为K
rate=biterr(msg, m sg r, M)
```

程序运行结果为:

```
rate =
    0
```

例5-8　在MATLAB中通过编程实现以下功能:产生0~8之间的随机整数序列,然后用8进制正交幅度键控(8-QASK)进行调制。

MATLAB程序如下:

```
%MATLAB program 5-8
%8-QASK
M = 8;                                          % 用字母表示的符号数
len = 10000;                                    % 初始化符号数
Fd = 1;                                         % 假设初始化采样率每秒为1
Fs = 3;                                         % 被调制信号的采样率每秒为3
signal = randint(len,1,M);                      % 随机产生整数序列产生的
                                                % 信号是由0到M-1组成
modsignal(:,1) = dmodce(signal,Fd,Fs,'qask',M); % 使用8-QASK进行调制
                                                % 两个标定不同的正方形星座
inphase = [-3:2:3, -3:2:3];
quad = [ones(1,4), -1 * ones(1,4)];
modsignal(:,2) = dmodce(signal,Fd,Fs,'qask/arb',inphase,quad);
noisy = modsignal + .5 * randn(len * Fs/Fd,2)...
   +j * .5 * randn(len * Fs/Fd,2);
newsignal(:,1) = ddemodce(noisy(:,1),Fd,Fs,'qask',M);   % 解调使恢复信号
newsignal(:,2) = ddemodce(noisy(:,2),Fd,Fs,...
   'qask/arb',inphase,quad);
[num,rate] = biterr(newsignal,signal);          % 比较新信号与原信号的每一列
disp('Bit error rates for the two constellations used here')
disp('------------------------')
disp(['Gray code constellation:      ', num2str(rate(1))])
disp(['Non-Gray code constellation: ', num2str(rate(2))])
modmap('qask',M);                               % 画出由格雷码标示的信号星座图
```

星座图如图5-43所示。

程序运行结果为:

```
Bit error rates for the two constellations used here
--------------------
Gray code constellation:      0.00016667
Non-Gray code constellation: 0.0001
>> Bit error rates for the two constellations used here
--------    ---------
Gray code constellation:      0.0001
```

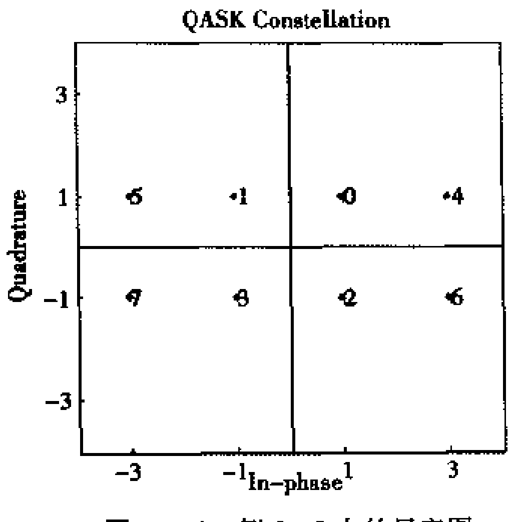


图 5-43　例 5-8 中的星座图

Non-Gray code constellation: 0.00046667

眼图是研究码间干扰和信道衰减的一种既简单又方便的实用工具，使用眼图的一种方法就是观察"眼睛"睁开最大的地方，把此点作为判决点。生成信号的眼图可以使用 eyediagram 函数。下面通过一个例子说明如何生成信号的眼图和从眼图中找出最佳判决点。

例 5-9　将一随机数字信号映射为 16-QASK 波形，用升余弦滤波器仿真噪声传输信道，然后用 eyediagram 命令生成信号的眼图。

创建信号眼图的 MATLAB 程序如下：

```
%MATLAB program 5-9
% Produce eye diagram
clear all;
clc;
M = 16; Fd = 1; Fs = 10;                          % 定义所采用的进制数和采样率
Pd = 100;                                         % 计算所采用的点数
delay = 3;
msg_d = randint(Pd,1,M);                          % 在[0,M-1]区间产生随机整数
msg_a = modmap(msg_d,Fd,Fd,'qask',M);             % 用矩形星座 QASK 方法进行调制
rcv = rcosflt(msg_a,Fd,Fs,'fir/normal',.5,delay); % 将信道近似为升余弦滤波器，且升
                                                  % 余弦滤波器的延迟时间定为 3 秒
propdelay = delay .* Fs/Fd + 1;                   % 截去升余弦滤波器输出的尾部响应
rcv1 = rcv(propdelay:end-(propdelay-1),:);
N = Fs/Fd;
offset1 = 0;                                      % 绘出信号的眼图并显示出没有偏移
                                                  % 的情况
h1 = eyediagram(rcv1,N,1/Fd,offset1);
set(h1,'Name','Eye Diagram Displayed with No Offset');
offset2 = 2;                                      % 绘出信号的眼图并显示出存在偏移
                                                  % 的情况
h2 = eyediagram(rcv1,N,1/Fd,offset2,'r-');
```

```
set(h2,'Name','Eye Diagram Displayed with Offset of Two');
```

程序运行结果如图5-44和图5-45所示。

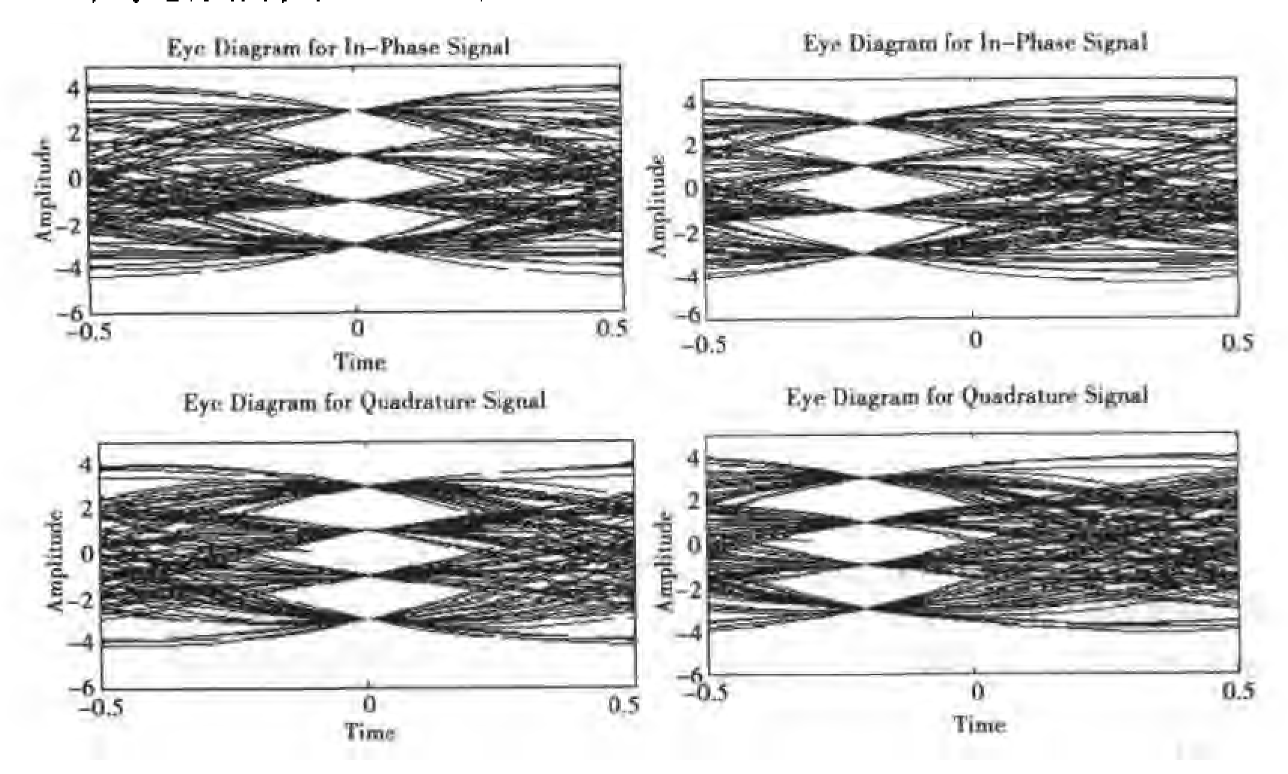


图5-44　例5-9中无偏移情况下的眼图　　　图5-45　例5-9中有偏移情况下的眼图

注意：在图5-44中，眼图中心的垂直线处是"眼睛"睁开最大的地方，这是无偏移的情况；在图5-45中，眼图中"眼睛"睁开最大的地方在中心偏左处，这是有偏移的情况。

5.5　本章小结

本章主要介绍了MATLAB在通信系统仿真中的应用。首先介绍了通信系统的一般模型和通信系统的仿真模型，在此基础上，重点介绍了在SIMULINK环境下进行通信系统仿真所需的各种仿真模块和用数据流仿真方法进行仿真所需的各种通信系统仿真函数；然后通过众多实例详细说明了在MATLAB中如何进行通信系统的仿真。

习　题

1. 用时间流仿真的方法产生一维和二维服从瑞利分布的信号，并用显示器显示出来。试用同样的方法产生2×3的矩阵信号。
2. 绘制并观察正弦波信号的眼图和散布图。

3. 在工作空间产生一个指数信号，并将此信号作为信号源，采用图 5－46 的仿真图所示的方法对信号进行量化。

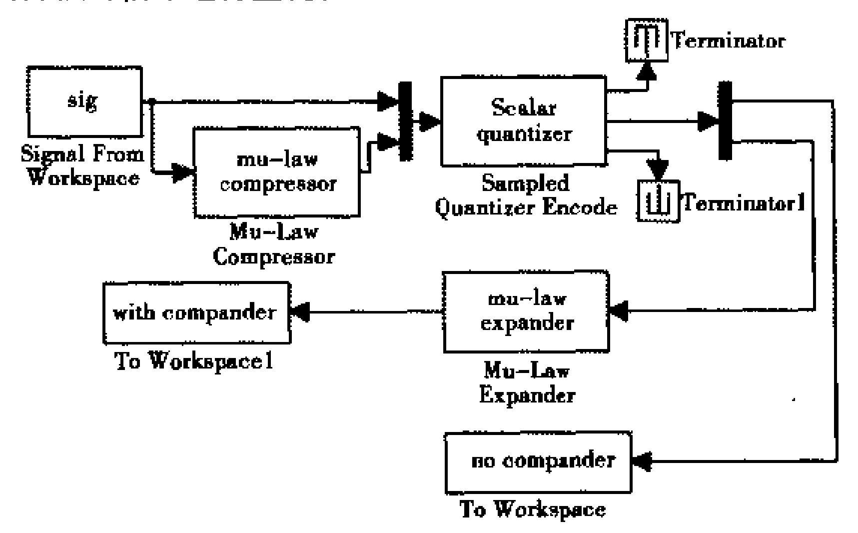


图 5－46　习题 3 的仿真图

4. 采用 DQPSK 调制方法对随机整数信号进行调制，绘制并观察其相位图。
5. 用随机整数信号源产生器产生信号，用矩阵 QAM 调制法进行调制，信道采用 AWGN 信道，绘制接收信号的眼图和散布图。
6. 采用数据流的方法对习题 3 中的仿真模型进行仿真。
7. 使用 M 文件编程实现习题 5 中的通信系统。
8. 实现下列通信系统的仿真：
   - 伯努利随机二进制信号源产生器。
   - 采用海明编/译器进行编码和译码。
   - AWGN 信道。

   计算出由噪声导致的传输错误率，并想办法降低这种错误率。

第6章　MATLAB在电子电路仿真中的应用

知识点：

- 模拟电路仿真
- 数字电路仿真

本章首先介绍在MATLAB的SIMULINK环境下进行模拟电路和数字电路仿真所需的仿真模块和仿真命令；然后介绍如何在SIMULINK环境下创建模拟电路系统和数字电路系统的模型；最后介绍模拟电路系统与数字电路系统的仿真方法与技巧。

通过本章的学习，读者可以了解MATLAB中有关模拟电路和数字电路分析与设计、仿真所需要的各种仿真模块和仿真命令，以及如何利用这些模块和命令进行复杂电路系统的仿真。本章的内容对于教师讲授电子电路方面的课程，学生学习电子电路方面的知识和工程师从事电子电路设计都会有事半功倍的效果。

从本书第4章和第5章可以看出，利用MATLAB仿真技术对信号和系统进行分析、设计，对数字信号进行处理，不仅可以快速直观地查看分析和设计结果，还可以反复地进行修改、验证，从而大大节省设计时间，节省人力、物力和财力。在MATLAB中，除提供了信号处理和通信系统工具箱外，还提供了专门用于进行各种模拟电路系统和数字电路系统仿真的SIMULINK软件包。下面将介绍如何利用MATLAB仿真技术进行模拟电路和数字电路的分析和设计。

6.1　模拟电路仿真

在MATLAB 5.2以上版本的SIMULINK中，均有一个由加拿大的HydroQuebc公司和TECSIM International公司共同开发的电力系统仿真模块库（Power System Blockset，PSB），其功能非常强大，可用于对电路系统、电力电子系统、电机系统、电力传输过程等进行仿真。这个工具箱提供的是一种类似电路建模的方式进行模型绘制，在仿真前自动将其转化成状态方程描述的系统形式，然后再在SIMULINK环境下进行仿真。

由于Power System Blockset功能非常强大，内容十分丰富，这里不可能对其中的所有模块和命令都进行详细地探讨，所以本节将主要讨论在进行模拟电路系统的分析与设计过程中所需的主要模块和命令，以及它们的使用方法。而对于在模拟电路系统仿真中不需要的模块和命令只进行简要介绍。

6.1.1　仿真模块与应用技巧

1. 仿真模块与技巧

在 MATLAB 命令窗口中执行 powerlib 命令，或在 SIMULINK 环境下的模块库窗口中用鼠标左键单击 Power System Blockset(电力系统仿真模块库)，都将会弹出如图 6-1 所示的 Power System Blockset 模块库。

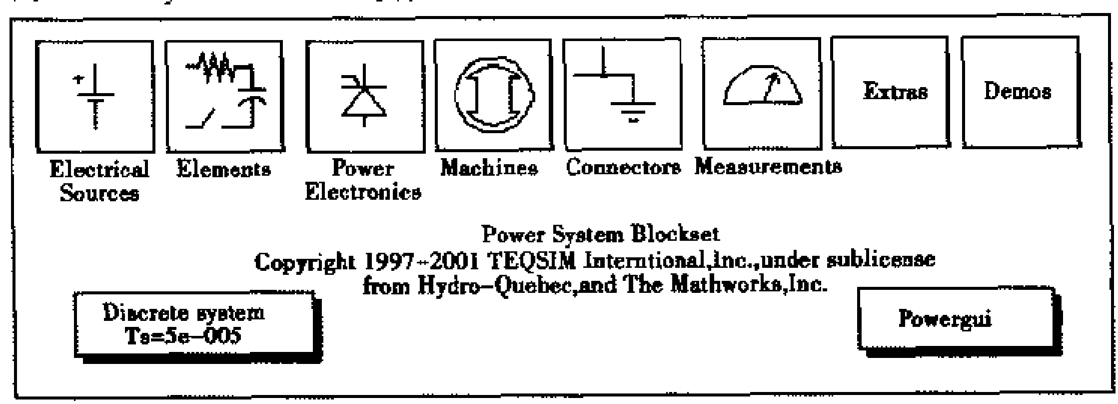


图 6-1　电力系统模块库

在 Power System Blockset 模块库中，共包括 10 个模块，其中有 7 个是仿真子模块集，在每个子模块集中又包含若干个仿真模块，用鼠标双击各个子模块集，便会打开相应的子模块集，从中可以得到所需的模块。下面对 Power System Blockset 模块库中的模块集进行介绍。

◆　**Electrical Sources(电源子模块集)**

如图 6-2 所示，Electrical Sources(电源子模块集)中包含 DC Voltage Source(直流电压源)、AC Voltage Source(交流电压源)、AC Current Source(交流电流源)、Controlled Voltage Source(可控电压源)和 Controlled Current Source(可控电流源)等 5 个模块。

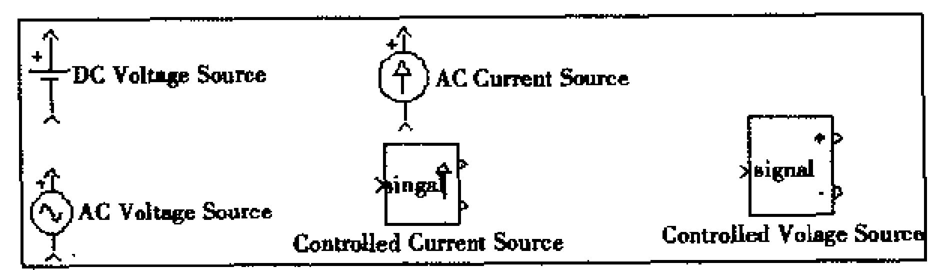


图 6-2　电源子模块集

◆　**Elements(电路元件子模块集)**

如图 6-3 所示，Elements(电路元件子模块集)中包含 Series RLC Branch(串联 RLC 分支)、Series RLC Load(串联 RLC 负载)、Parallel RLC Branch(并联 RLC 分支)、Parallel RLC Load(并联 RLC 负载)、线性变压器、饱和变压器、互感器、电涌放电器、分布参数线路、Breaker(断路器)以及 PI Section Line(π 截面导线模块)、Three-phase blocks(三相模块)和 Three-Phase Transformer(三相变换器模块)等模块。

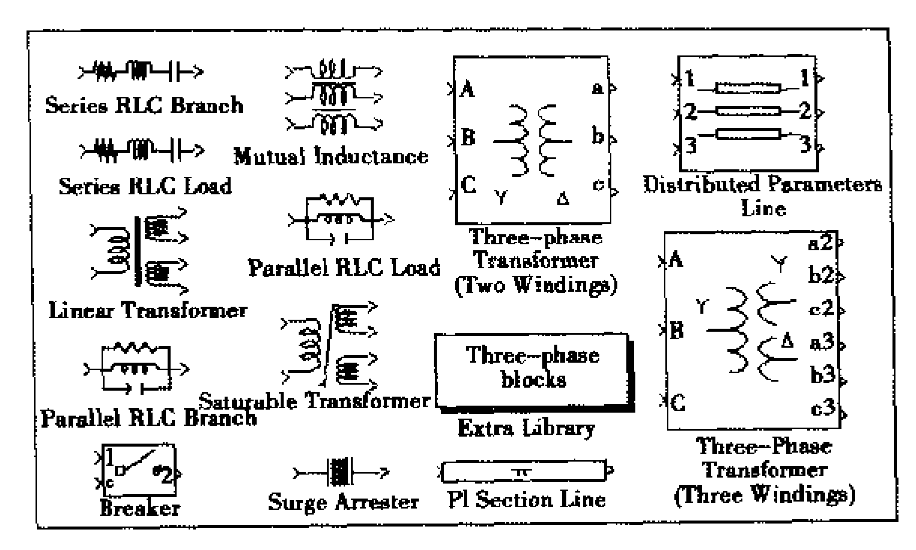


图 6-3　电路元件子模块集

需要说明的是，在电路系统的仿真中，电阻、电容及电感元件是最常用的电路元件，但电路元件子模块集中没有这三个电路元件。不过，单独的电阻、电容及电感元件可由串联或并联的 RLC 仿真模块修改而得到，但这种修改过程不是直接将元件的参数置为 0，而是需要在参数设置窗口中将 L 和 C 的参数设置改为 0 和 inf(无穷大)，将 R 的参数设置为需要的电阻值。在进行参数设置时一定要注意电路参数的单位，R、L 和 C 的单位分别为 Ω(欧)、H(亨)和 F(法)。另外，为了方便创建模型，也可以拆出单个电路元件封装成子系统放入模块库中，以备调用。

表 6-1 为单个电阻、电容、电感参数设置表。

表 6-1　单个电阻、电容、电感参数设置表

元件类型	串联 RLC 模块			并联 RLC 模块		
	电阻参数	电感参数	电容参数	电阻参数	电感参数	电容参数
单个电阻	R	0	0	R	INF	INF
单个电容	0	0	C	0	0	C
单个电感	0	L	0	INF	L	INF

下面的例 6-1 说明了如何由 RLC 串联和并联仿真模块构造单独的电阻、电容和电感元件。

例 6-1　用 SIMULINK 中的模块绘制一个由电压源、电阻、电容和电感组成的简单电路，如图 6-4 所示。

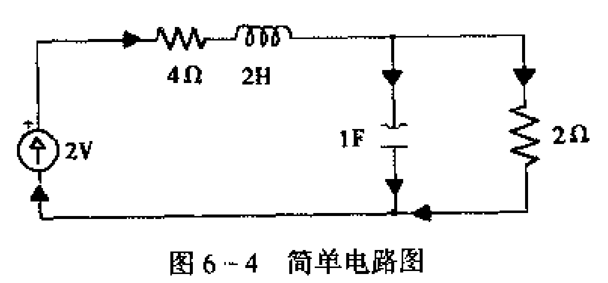


图 6-4　简单电路图

◆ **Power Electronics(功率电子仿真子模块集)**

如图 6-5 所示,Power Electronics(功率电子仿真子模块集)中包括 Ideal Switch(理想开关)、Diode(二极管)、Detailed Thyristor(晶闸管)、Gto(可关断可控硅)、Mosfet(场效应管)、Universal Bridge(电桥)、IGBT(绝缘栅二极管)模块和 Control blocks(控制模块)与 Discrete Control blocks(离散控制模块)两个扩展子模块集。

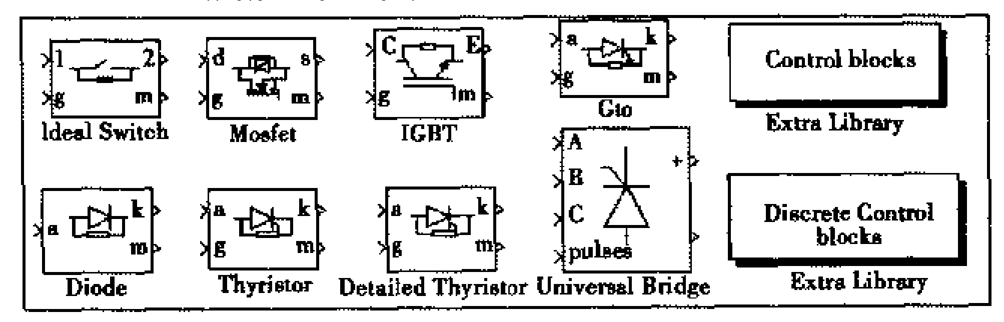


图 6-5 功率电子仿真子模块集

◆ **Measurements(测量仪表子模块集)**

如图 6-6 所示,Measurements(测量仪表子模块集)中的模块主要用于电路参数的测量,其中包括 Current Measurement(电流表)、Voltage Measurement(电压表)、Impedance Measurement(阻抗表)、电流电压表 4 个仿真模块和 Measuement blocks(测量模块)与 Discrete Measuement blocks(离散测量模块)两个扩展子模块集。

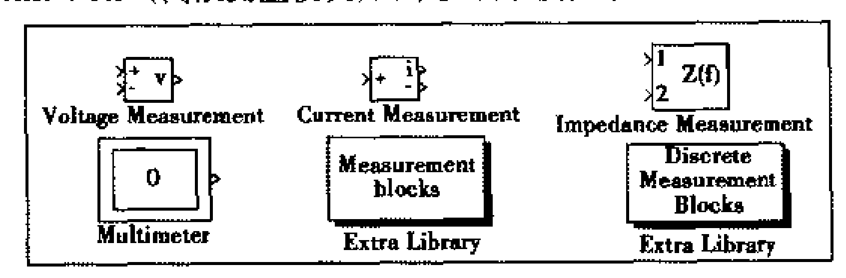


图 6-6 测量仪表子模块集

◆ **Connectors (连线器子模块集)**

在一般电路中,除了前面介绍的各种电路元件外,还需要一些连线器类的模块,用于实现各电路元件的连接。如图 6-7 所示,连线器子模块集中含有输入输出模拟接地、输入输出数字接地、T 型和 L 型连接器、水平垂直的多进多出连接器以及水平垂直的多进多出薄连接器。

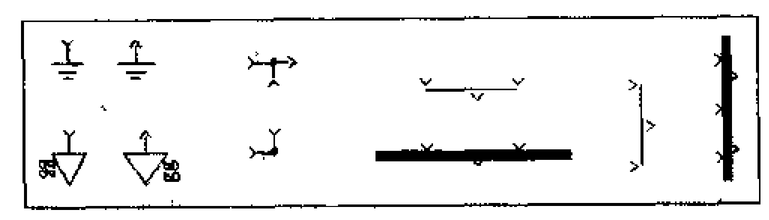


图 6-7 连线器子模块集

在 Power System Blockset 模块库中除了包括上述 5 个子模块集外,还包括 Machines(电机子模块集)、Extras(附加子模块集)、Discrete System(离散采样时间模块)、MATLAB 实例演示窗口的接口模块以及电源图形用户界面模块等模块。

提示：上面介绍的只是电路中的主要模块，在实际的电路系统仿真中，还需要调用基本函数模块(如求模、求相位等模块)、输出模块(如示波器、XY 坐标等)。

2. 仿真命令

电力系统工具箱中提供的函数 power2sys()可以用于提取从给定电源到输出端子的状态方程模型，根据此状态方程模型就可以对整个电路进行频域分析，该函数的调用格式为：

[a,b,c,d] = power2sys('模型名')

其中，a、b、c、d 为系统的状态方程矩阵。

6.1.2 仿真方法与应用实例

下面通过实例详细介绍模拟电路的仿真模块和仿真命令的使用方法，以及如何实现这些仿真模块、命令与本书前面章节所讲内容(如信号源库、接收模块库、函数库等)的有机结合。

例 6-2 介绍了如何运用 MATLAB 仿真技术进行简单电路的分析和简单电路元件及电压、电流测量表的使用方法。

例 6-2 求如图 6-8 所示的电路的各支路的电流和电压。

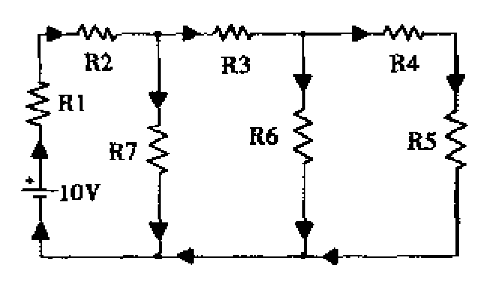


图 6-8 例 6-2 的电路图

在求解这道题时，一般的解法是根据电路图用电路分析方法(如节点分析法、网孔分析法等)列出电压或电流方程，再求解出各支路的电压和电流，或者在求解的过程中利用 MATLAB 的科学计算功能辅助求解。显然，其求解过程是很复杂的，有没有更简单的方法可以求解呢？答案是肯定的。如果将电路图用元器件和导线连接起来，再用电压表和电流表测量，结果就很容易得到，而且相当准确。但是，用真实的电器元件进行连接显然是不现实的。而 MATLAB 的 SIMULINK 动态仿真技术为此提供了便利的模拟工具。

首先将上述电路图用 MATLAB 提供的电路仿真模块进行连接，并将各电路元件的参数按图中的值进行设置；然后将电压表和电流表连接到电路中去，这时便可以进行电压和电流的测量。电压表和电流表的连接方法如图 6-9 所示。

这样，要求哪一支路的电压或电流，只需要将电压表或电流表连接到相应的支路，然后通过示波器就可以直接观察到其电压或电流值。为叙述方便，这里将电路图中的各支路用它们电阻的标号来代表，如电阻 R1 的支路称为支路 1，电阻 R2 的支路称为支路 2，其余类推。于是可得三个环路的电流值(如图 6-10 所示)和 7 个支路的电压值(如图 6-11 所示)。图 6-10 中电流值的单位为安(A)，图 6-11 中电压值的单位为伏(V)。

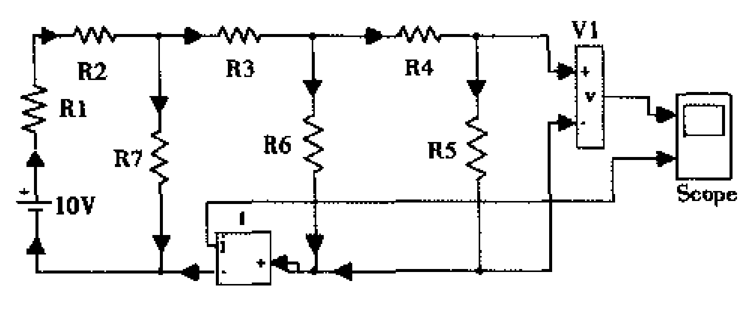


图6-9 电压表和电流表的连接方法

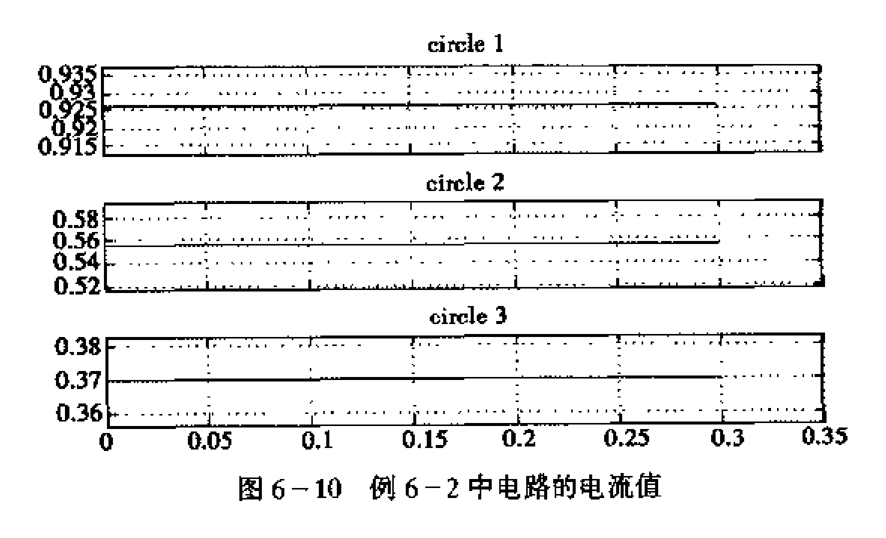


图6-10 例6-2中电路的电流值

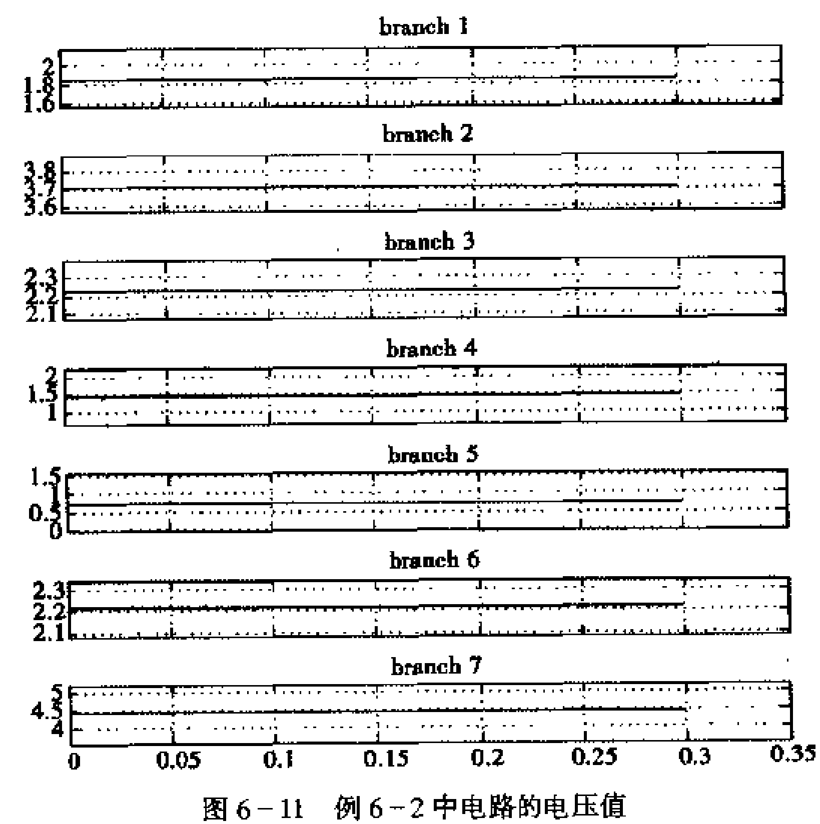


图6-11 例6-2中电路的电压值

提示:MATLAB提供的是一种类似电路建模的方式进行模型绘制,在仿真前先自动将其转化成状态方程描述的系统形式,然后才在SIMULINK环境下进行仿真。因此当

运行仿真时，在 MATLAB 命令窗口会出现以下语句，以表示创建的模型是否正确，仿真是否进行完毕。

Power System Blockset processing circuit ...

Ready.

若用命令在 MATLAB 命令窗口进行仿真，可以键入以下语句：

```
[a,b,c,d] = power2sys('circuit')
```

运行后得到的结果为：

```
a =      []
b =      Empty matrix: 0 - by - 1
c =      Empty matrix: 7 - by   0
d =
    0.0741
    0.3704
    0.1852
    0.2222
    0.1481
    0.2222
    0.4444
```

当分析直流电源的电路时，采用代数方法即可进行分析，但当电源为交流电源时，则需要采用相量或复数的方法进行求解。而采用相量法分析电路是非常复杂的事，这时若采用 MATLAB 仿真技术，则可大大简化其求解过程。下面通过实例说明由电阻、电容和电感组成的交流电路的仿真方法与仿真技巧。

例 6-3 分析图 6-12 所示的电路，求出各支路的电流和电压波形。

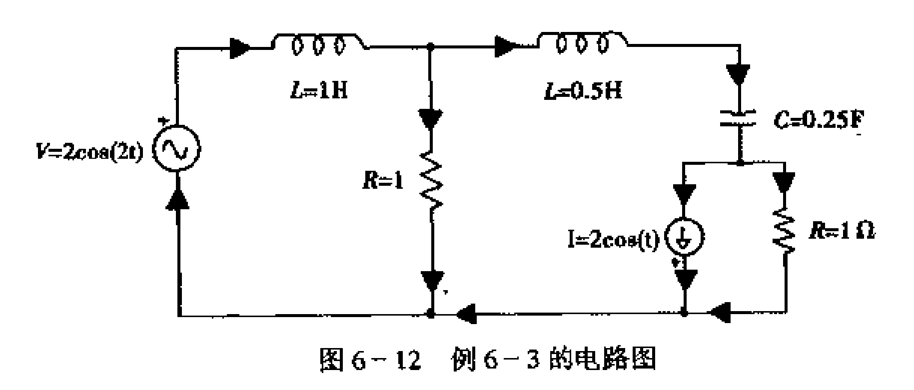


图 6-12 例 6-3 的电路图

首先按照图 6-12 所示的电路图在 MATLAB 的 SIMULINK 环境下建立好电路的模型，电路中各元器件的参数设置如表 6-2 所示。

表 6-2 例 6-3 中各元器件的参数设置表

电路元件参数	$L=1$H	$L=0.5$H	$R=1$	$R=1\Omega$	$C=0.25$F
电阻 $R(\Omega)$	0	0	1	1	INF
电感 L(H)	1	0.5	0	0	INF
电容 C(F)	INF	INF	INF	INF	0.25

然后求得各支路的电压和电流的波形。这时，必须在电路中加入电压、电流测量表，加入后的电路图如图6-13所示。

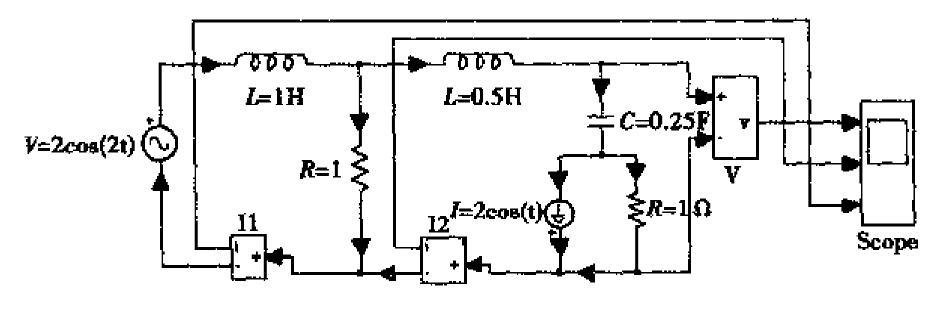


图6-13 例6-3电路的测量图

电路仿真模型运行后，在MATLAB命令窗口中将显示：

```
Power System Blockset processing circuit2 ...
Computing state-space representation of linear electrical circuit (V2.2)...
(3 states ; 2 inputs ; 3 outputs)
Computing steady-state values of currents and voltages ...
Build the SIMULINK equivalent circuit ...
(Circuit stored inside "circuit2/V" block)
Ready.
```

各支路的电压和电流的波形如图6-14所示。

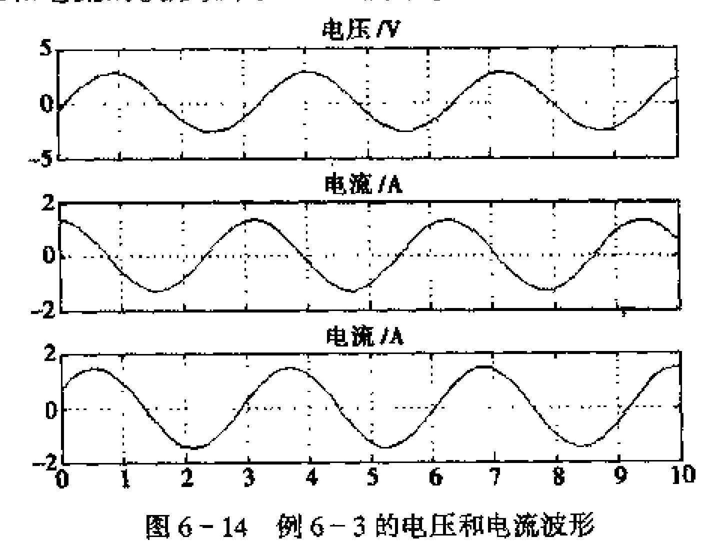


图6-14 例6-3的电压和电流波形

若用命令在MATLAB命令窗口进行仿真，可以键入下列语句：

```
[a,b,c,d]=power2sys('circuit2')
```

运行后的结果为：

```
a =
    -4     2    -2
     1    -1     0
     4     0     0
b =
     0     2
```

```
        1       0
        0       0
c =
        1       0       1
        0       1       0
        1       0       0
d =
        0      -1
        0       0
        0       0
```

6.2 数字电路仿真

到目前为止，MATLAB 虽然还没有提供用于数字电路仿真的专用工具箱，但在数字电路的学习和教学中，若能较好地利用 MATLAB 仿真技术，可以使教师的讲解更生动、更容易被学生理解，同时可以使学生能形象直观地看到电路运行时的波形变化，可以大大提高学习效率。另外，优化和调试数字电路、验证数字电路正确与否是一件十分繁琐的工作，工程师们通常的做法是：用面板搭接电路，或者将元器件焊接到预先设计好的印制板电路上进行测试、修改、完善。前一种方法往往由于连线多易造成连线错误，因接触不良而造成功能失常；后一种方法则往往由于设计、焊接、调试、改线的多次反复而花费大量的时间、精力。可见，要完成这样的工作既费时又费力，而且还需要花费大量的资金去添置各种工具和仪器，若能采用 MATLAB 仿真工具 SIMULINK 进行数字电路的调试、仿真、验证，则可以避免上述两种方法的缺点，不仅省时、省力，而且还可以节约大量的成本。

6.2.1 仿真模块与应用技巧

1. 基本数字电路模块

SIMULINK 提供的用于数字电路仿真的常用模块包括信号源、输出设备、触发器和寄存器。下面分别对它们进行介绍。

◆ **Signal Sources(信号源)**

Signal Sources(信号源)包括 CLOCK(时钟)、Step(阶跃信号)、Pulse Generator(脉冲发生器)、Signal Generator(信号发生器)等，如图 6－15 所示。

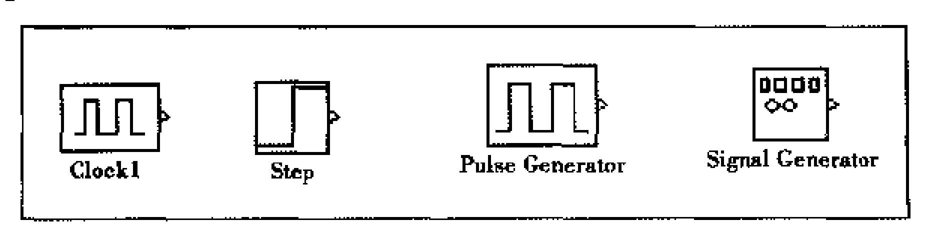


图 6－15　信号源模块

◆　**输出设备**

输出设备包括 Scope(示波器)、XY Graph(XY 坐标图)、To File(输出到文件)和 To Workspace(输出到工作空间)等,其中用得最多的输出设备是示波器。

用 Scope(示波器)观察信号有多种方法和技巧,读者可以用游离示波器观察信号,其方法是:首先将示波器悬空,设置好示波器参数;然后用鼠标单击某模块间连线,再在游离示波器窗口的菜单栏中执行 Simulation/Start 命令进行仿真,此时示波器中所显示波形即为该连线上的信号波形。在仿真过程中,如果想观察另一连线上的信号,可以用鼠标单击另一连线使其显示出小黑方块句柄即可。另外,还可以用一个示波器观察多路信号,只需将多路信号通过多路开关(Mux)引到示波器中即可。在仿真过程中,如果想仔细观察某一段波形,可以用鼠标单击这段波形的开始处,按着鼠标不放,同时拖动鼠标斜拉出一个矩形,这时在该矩形区域内的波形将被放大。

◆　**Flip-Flops(触发器)**

在数字电路仿真中,常用的时序电路模块在 SIMULINK Extras 模块库的 Flip-Flops 模块组中,这些模块为 D 触发器、R-S 触发器、J-K 触发器等,如图 6-16 所示。

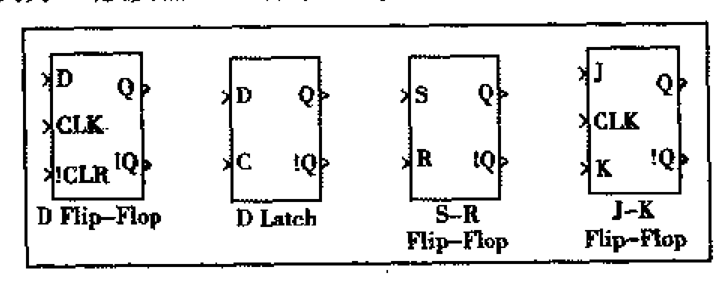


图 6-16　触发器模块

◆　**基本的数字逻辑运算模块和算术运算模块**

基本的数字逻辑运算模块和算术运算模块包括 AND(与)、OR(或)、NOT(非)、NAND(与非)、NOR(或非)、XOR(异或)、Add(加)、Subtract(减)、Multiply(乘)、Divide(除)等。基本的数字逻辑运算模块保存在 Fixed-Point 模块库中的 Logic & Comparison 模块组中,基本的算术运算模块保存在 Fixed-Point 模块库中的 Math 模块组中,如图 6-17 所示。

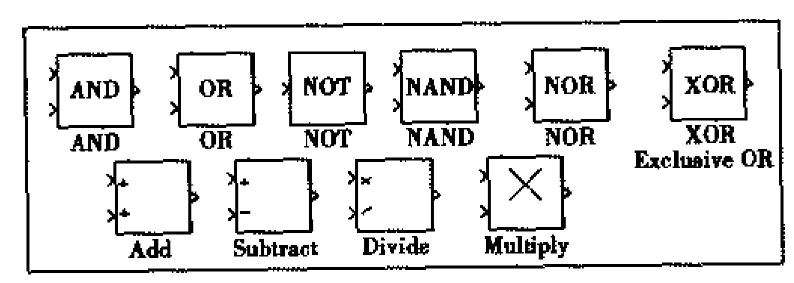


图 6-17　基本的数字逻辑运算模块和算术运算模块

提示:利用基本的数字逻辑运算模块和算术运算模块可以进行组合逻辑电路的设计,如果将基本的数字逻辑运算模块和算术运算模块与时序逻辑电路模块结合,则可以进行时序逻辑电路的设计。

2. 自定义数字电路模块

自定义功能模块有多种方法，其中的一种是将现有的多个功能模块组合起来，形成新的模块，这样可以简化图形，减少功能模块的个数。另外也可以建立自己的模块库，为以后搭接数字仿真电路提供方便。

在 Extras 模块库的 Flip-Flops 组中，包括 D 触发器、R-S 触发器、J-K 触发器等时序逻辑电路的基本单元，既可以通过这些触发器组成如寄存器、计数器等具有单一功能的器件，也可以通过这些触发器组成功能复杂的子系统。但 SIMULINK 提供的这些触发器具有一定的缺陷，在存在外部反馈时会形成代数环，这时需要加以改造才能完成仿真。下面以 D 触发器和 R-S 触发器为例说明如何完善这些触发器模块。

◆ **D 触发器**

实际的 D 触发器具有两个特点：一是在时钟脉冲到来之前它保持原来的状态不变；二是它具有一定的延迟。在数字电路的仿真中，通过分析仿真波形会发现，SIMULINK 提供的 D 触发器模块不具备实际 D 触发器所具有的第二个特点，因此需要在 Q 输出端加一个 Unit Delay(延迟单元)。使用添加了延迟单元后的 D 触发器模块就可以完成数字电路的仿真。另外，SIMULINK 提供的 D 触发器模块没有置位端，为此，要在 D 触发器模块的输入端加一个 OR(或逻辑)模块，以组成具有置位端的 D 触发器模块。可以将上述添加了 Unit Delay(延迟单元)和 OR(或逻辑)模块的 D 触发器模块封装起来，以组成自己的 D 触发器模块(子系统)。

D 触发器的封装过程为：首先从 SIMULINK 的模块库中选择 D 触发器、OR(或逻辑)模块、Unit Delay(延迟单元，此模块在 SIMULINK 下的 Discrete 中)，将它们按图 6－18 所示连接起来；然后在 SIMULINK 窗口的菜单栏中执行 Edit/Creat Subsystem 命令，这时 D 触发器模块就建成了。封装后得到的自定义 D 触发器模块如图 6－19 所示。

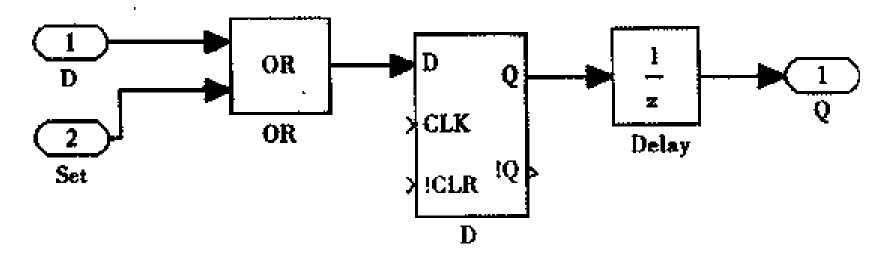


图 6－18　D 触发器子系统

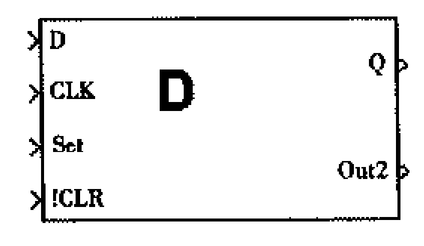


图 6－19　自定义 D 触发器

◆ **R-S 触发器**

对于 R-S 触发器，如果 S＝1、R＝0，则 Q 输出为 1；如果 S＝0、R＝1，则 Q 输出为 0；

如果 S=0、R=0,则 Q 保持原来的状态;而对于 S=1、R=1 的情况,则要尽量避免发生。SIMULINK 提供的 R-S 触发器由 Combinatorial Logic(逻辑真值表模块)、Mux(多路信号合成模块)、Demux(多路信号分解模块)和 Memory(寄存器模块)组成,如图 6-20 所示。

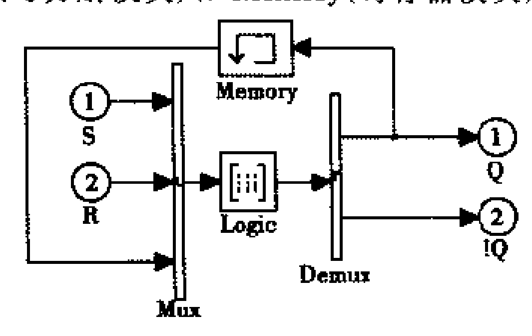


图 6-20　R-S 触发器

R-S 触发器也可以由两个 AND(与逻辑)及两个 NOR(或非逻辑)模块组成,如图6-21所示。为了避免产生代数环的错误,可以在反馈的地方加上两个加法器以产生初始值。另外,也可以在 R-S 触发器的前端加一个激活功能模块,使其成为具有使能端的 R-S 触发器。其方法为:首先从 Connections 模块库中将 Subsystem 功能模块复制到设计区域中。由于激活功能模块只能放到 Subsystem 的设计区域中,所以必须进入 Subsystem 的设计区域中进行设计。将 Enable(激活功能模块)复制到 Subsystem 的设计区域中进行调整后,再将设计好的 R-S 触发器模块复制到 Subsystem 的设计区域中。然后将 Enable(激活功能模块)与 R-S 触发器的输入端连接起来。最后将具有使能端的 R-S 触发器封装存档。

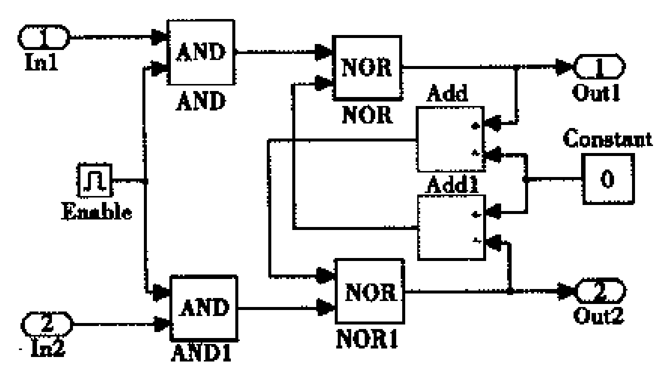


图 6-21　具有使能端的 R-S 触发器

6.2.2　仿真实例

在本章的前面内容中介绍了数字电路仿真的基本模块和自定义模块的相关知识与使用技巧。下面通过数字电路仿真实例,介绍 MATLAB 仿真技术在数字电路仿真中的强大作用,以使读者能较好地掌握在 MATLAB 中进行数字电路仿真的方法和技巧。

例 6-4 是组合逻辑电路的设计,通过对该实例的学习,读者可以学会如何运用 MATLAB 仿真技术进行组合逻辑电路的设计和仿真。

例 6-4　组合逻辑电路设计:试用数字电路仿真模块设计一个 2 线-4 线译码器。

2 线-4 线译码器的真值表如表 6-3 所示。

表 6－3　2 线－4 线译码器的真值表

A	B	F1	F2	F3	F4
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

由 2 线－4 线译码器的真值表可得到其逻辑表达式为：

$$\begin{cases} F1 = AB, & F2 = \overline{AB} \\ F3 = \overline{A}B, & F4 = \overline{AB} \end{cases}$$

根据逻辑表达式可画出译码器的逻辑电路仿真图，如图 6－22 所示。

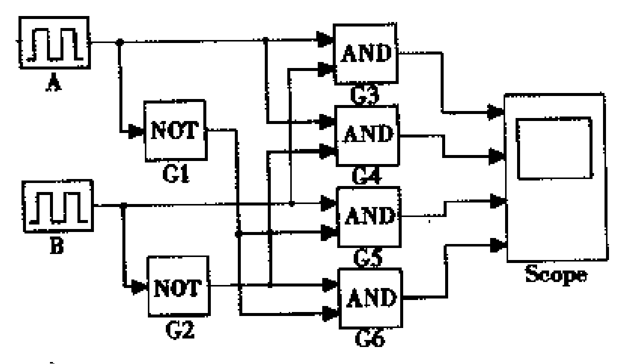


图 6－22　2 线－4 线译码器的逻辑电路图

说明：图 6－22 中逻辑门 G3、G4、G5 和 G6 的输出分别为 F1、F2、F3 和 F4，它们的输出波形如图 6－23 所示。

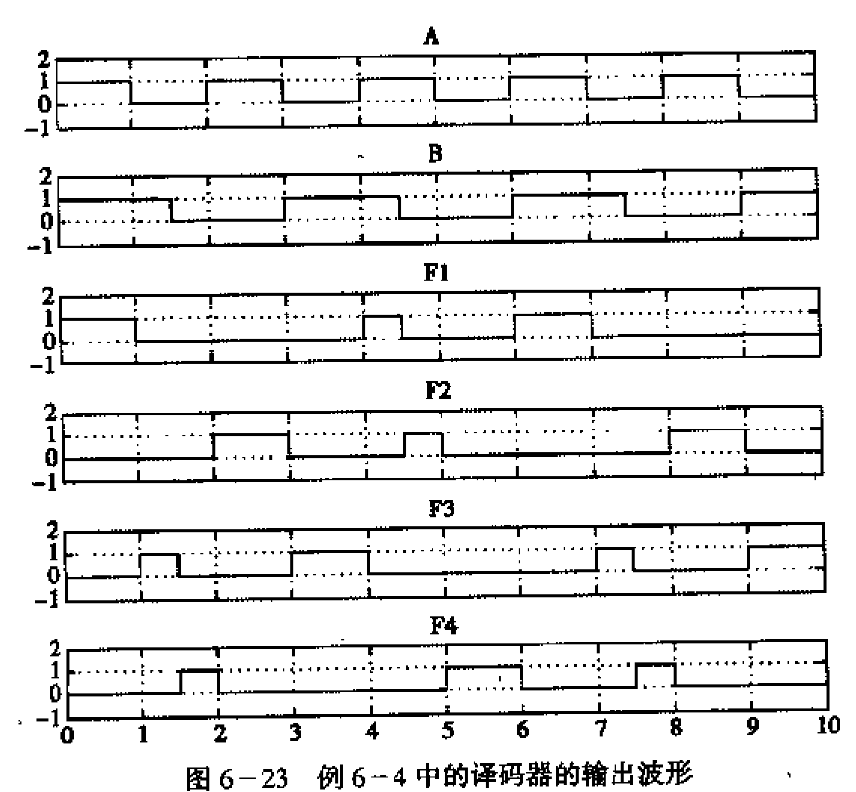


图 6－23　例 6－4 中的译码器的输出波形

例 6-5　检验自定义 D 触发器的正确性。

首先将前面封装好的自定义 D 触发器复制到 SIMULINK 仿真窗口中，将它与 Pulse（脉冲模块）、Clock（时钟模块）、Step（阶跃输入信号产生模块）连接起来组成仿真系统，如图 6-24 所示。

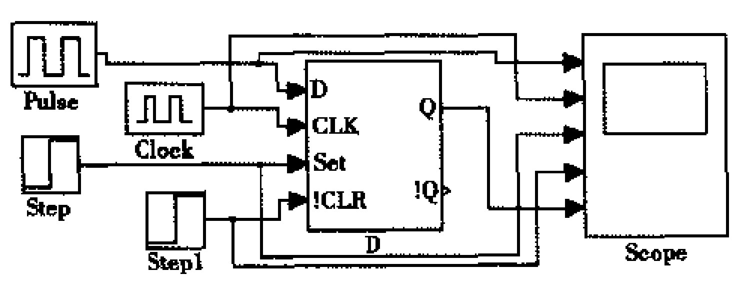


图 6-24　D 触发器仿真电路图

在图 6-24 中，Step（阶跃信号产生模块）的作用是置位，Step1（阶跃信号模块）的作用是清零，Scope（示波器）用来观察 D 触发器的输入、时钟、置位、复位和输出五路脉冲信号。仿真波形如图 6-25 所示。

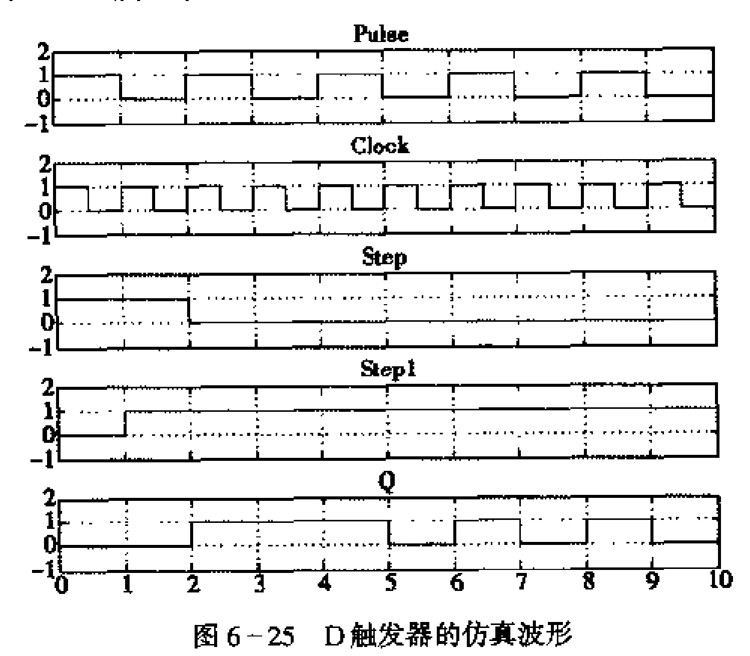


图 6-25　D 触发器的仿真波形

例 6-6　试设计一个七进制计数器。

由数字电路理论课中数字电路的设计方法可知，设计一个时序逻辑电路的过程应分为以下几个步骤：

◆ **逻辑抽象**

因为计数器的工作特点是在时钟信号操作下自动地依次从一个状态转为下一个状态，所以它没有输入逻辑信号，只有进位输出信号。可见，计数器属于摩尔型的一种简单时序电路。

取进位信号为输出逻辑变量 C，若分别用 $S_0, S_1, \cdots, S_6$ 表示，则按题意可画出如图 6-26所示的电路状态转换图。

因为七进制计数器必须用七个不同状态表示已经输入的脉冲数，所以状态已不能再化简。

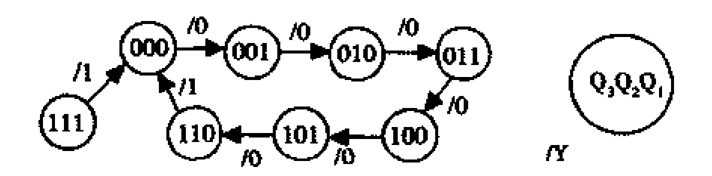


图 6－26　例 6－6 的电路状态转换图

◆　**状态编码**

由于计数器有七个状态，故触发器的位数应为 3，于是可得到如表 6－4 所示的电路状态编码表。

表 6－4　例 6－6 的状态编码表

状态顺序	状态编码			进位输出 C	等效十进制数
	Q_1	Q_2	Q_3		
S0	0	0	0	0	0
S1	0	0	1	0	1
S2	0	1	0	0	2
S3	0	1	1	0	3
S4	1	0	0	0	4
S5	1	0	1	0	5
S6	1	1	0	1	6
S7	0	0	0	0	0
S8	1	1	1	1	7
S0	0	0	0	0	0

◆　**写逻辑表达式及化简**

由表 6－4 所示的状态编码表，用卡诺图进行化简后得到简化的逻辑表达式。

$$\begin{cases} Q_1^{n+1} = \overline{Q_2^n Q_3^n}\, Q_1^n \\ Q_2^{n+1} = Q_1^n\, \overline{Q_2^n} + \overline{Q_1^n}\,\overline{Q_3^n}\, Q_2^n \\ Q_3^{n+1} = Q_1^n Q_2^n \overline{Q_3^n} + \overline{Q_2^n}\, Q_3^n \end{cases}$$

◆　**输出方程**

由表 6－4 所示的状态编码表，用卡诺图进行化简后得到简化的输出方程：

$$C = Q_2 Q_3$$

◆　**选择触发器及写出触发器的驱动方程**

这里选择 JK 触发器，根据 JK 触发器的特性方程可以写出三个 JK 触发器的驱动方程：

$$J_1 = \overline{Q_2^n Q_3^n} \quad K_1 = 1$$
$$J_2 = Q_1^n \quad K_2 = \overline{\overline{Q_1^n}\,\overline{Q_3^n}}$$
$$J_3 = Q_1^n Q_2^n \quad K_3 = Q_2^n$$

◆　**画出逻辑电路图**

用前面介绍的数字电路仿真模块，根据 JK 触发器的驱动方程和输出方程，画出七进

制计数器的逻辑电路图，如图 6－27 所示。

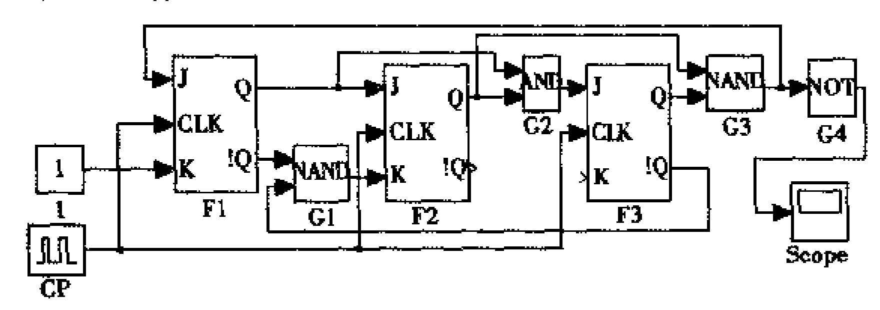


图 6－27　七进制计数器的逻辑电路图

◆ **逻辑功能验证**

当设计完了七进制计数器的逻辑电路后，还必须进行逻辑功能的校验。验证数字电路正确与否是一件十分繁琐的工作，传统方法是在设计过程中反复验证其设计思路和设计过程是否正确，如果掌握了 MATLAB 仿真技术，这项繁琐的工作就可以交给 MATLAB 完成，既简单快速又准确。

根据所设计的七进制计数器的逻辑电路，可用示波器观察其输出是否正确。七进制计数器的逻辑电路的仿真波形如图 6－28 所示。

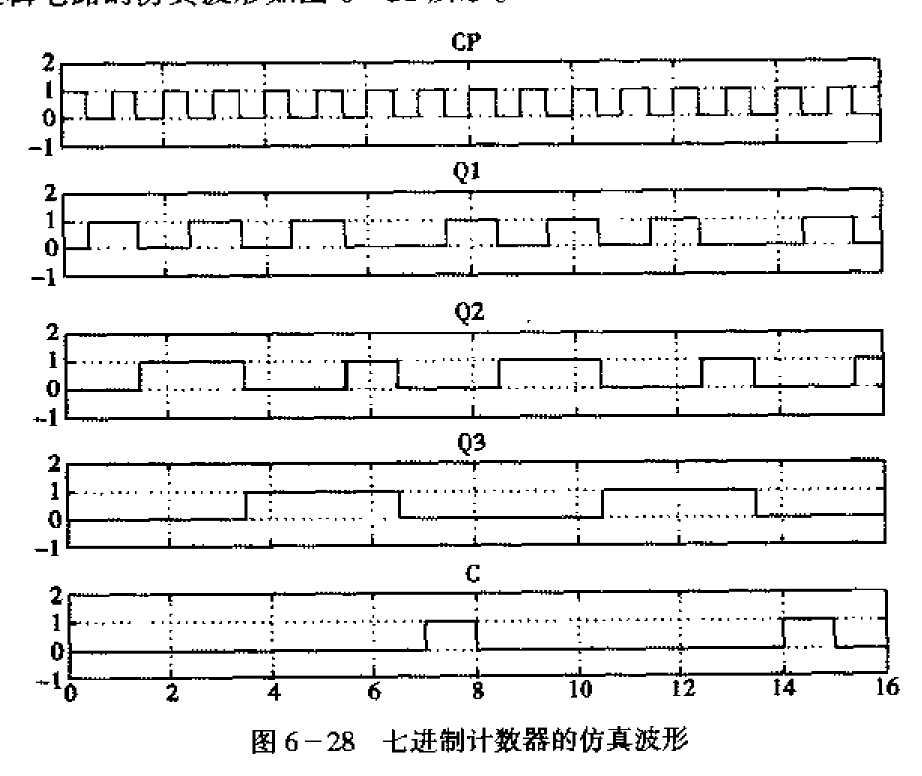


图 6－28　七进制计数器的仿真波形

6.3 本章小结

本章主要介绍了 MATLAB 仿真技术在模拟电路和数字电路仿真中的应用。首先介绍了模拟电路仿真所需要的各种仿真模块和仿真命令；然后通过众多模拟电路仿真实例介绍了模拟电路的仿真方法与技巧；最后通过众多数字电路仿真实例介绍了数字电路仿

真的仿真模块及仿真方法与技巧。

习　　题

1. 在 SIMULINK 环境下分别用串联型 RLC 和并联型 RLC 制作独立的电阻、电感和电容元件。
2. 在 SIMULINK 环境下用仿真方法求出下列电路中各支路的电流和电压值。

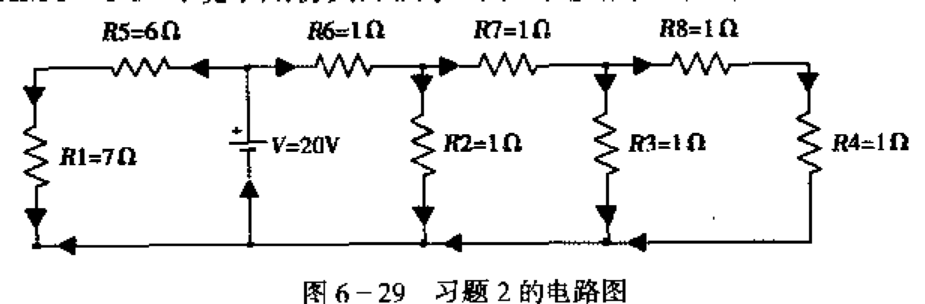


图 6－29　习题 2 的电路图

3. 用电路验证习题 1 中各独立电路元件的正确性。
4. 用仿真方法求出下面电路中各支路的电流和电压波形，并将它们用解析式形式表示出来。

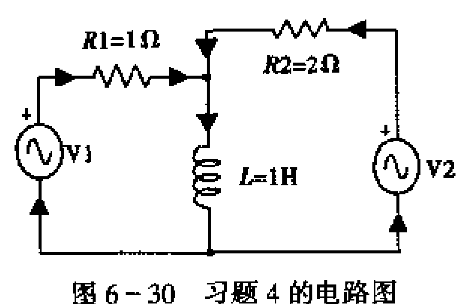


图 6－30　习题 4 的电路图

5. 用组合逻辑电路仿真模块设计一个能实现一位二进制相加的加法器。
6. 试设计一个 8 线－3 线的编码器电路。
7. 制作自己的 D 触发器。
8. 用习题 7 中自己制作的 D 触发器设计一七进制计数器，并验证设计结果是否正确。
9. 试用上升沿触发的 D 触发器设计异步二进制加法计数器和减法计数器。

第 7 章 MATLAB 在控制系统仿真中的应用

知识点：

- 控制系统模型
- 控制系统仿真
- 控制系统仿真实例

本章将主要介绍如何运用 MATLAB 仿真技术进行控制系统的仿真，内容包括系统数学模型的建立、系统的性能指标，以及控制系统仿真所需要的仿真模块、仿真命令、仿真模型的建立和控制系统的仿真方法。通过本章的学习，读者可以对控制系统的仿真有一个全面的了解，并学会在控制系统研究、分析和设计中灵活运用 MATLAB 仿真技术。

本章所介绍的控制系统是狭义的概念，特指自动控制系统。控制系统仿真是指以控制系统的模型为基础，主要用数学模型代替实际的控制系统，以计算机为工具，对控制系统进行实验和研究的一种方法。一般来说，控制系统仿真过程可以分为以下四个阶段：控制系统数学模型的建立、控制系统仿真模型的建立、控制系统仿真程序的编写和控制系统仿真实验及结果分析。目前，MATLAB 是进行控制系统仿真的最佳工具，因为 MATLAB 提供了控制理论及控制系统经常要遇到的复杂计算，如数组、向量、复数和矩阵等。

7.1 控制系统模型

7.1.1 数学模型

数学模型是计算机仿真的基础。数学模型是指描述系统内部各物理量（或变量）之间关系的数学表达式。控制系统的数学模型通常是指动态数学模型，自动控制系统最基本最重要的数学模型是输入输出模型，包括时域的微分方程、复数域的传递函数和频率域中的频率特性。除了输入输出模型之外，表示控制系统的数学模型还有状态空间模型、结构图模型等。这里以线性定常连续系统为例来介绍。

1. 输入输出模型

输入输出模型是指用系统的输入、输出信号或其变换式所表示的数学模型。当输入、输出信号为时域信号 $x(t)$、$y(t)$时，建立的数学模型是微分方程；当输入输出信号为复数域信号 $X(s)$、$Y(s)$（$X(s)=L[x(t)]$, $Y(s)=L[y(t)]$）时，建立的数学模型是传递函数；当输入输出信号为频率域信号 $X(j\omega)$、$Y(j\omega)$（$X(j\omega)=F[x(t)]$, $Y(j\omega)=F[y(t)]$）时，建立的数学模型是频率特性。下面分别给出这三种数学模型。

◆ **时域中的数学模型——微分方程**

微分方程是利用系统的物理定律来获取描述系统动态特性的数学模型。一般情况下，描述系统的线性常系数微分方程可表示为：

$$
\begin{aligned}
a_n \frac{\mathrm{d}y^n(t)}{\mathrm{d}t^n} + a_{n-1}\frac{\mathrm{d}y^{n-1}(t)}{\mathrm{d}t^{n-1}} + \cdots a_1 \frac{\mathrm{d}y(t)}{\mathrm{d}t} + a_0 y(t) \\
= b_m \frac{\mathrm{d}x^m(t)}{\mathrm{d}t^m} + b_{m-1}\frac{\mathrm{d}x^{m-1}(t)}{\mathrm{d}t^{m-1}} + \cdots b_1 \frac{\mathrm{d}x(t)}{\mathrm{d}t} + b_0 x(t)
\end{aligned}
\tag{7-1}
$$

其中，$a_i(i=0,1,\cdots,n)$，$b_j(j=0,1,\cdots,m)$均为实数，而且由系统本身的结构参数决定。

◆ **复数域中的数学模型——传递函数**

用微分方程表示的系统很难模拟成方框图，因而采用拉氏变换将输入、输出和系统表示成简单的代数关系。将(7－1)式进行拉氏变换并假设初始条件为0，就可以得到系统的传递函数模型：

$$
\frac{Y(s)}{X(s)} = G(s) = \frac{b_m s^m + b_{m-1}s^{m-1} + \cdots + b_0}{a_n s^n + a_{n-1}s^{n-1} + \cdots + a_0} \tag{7-2}
$$

在 MATLAB 中，如果知道了分子分母多项式，可以通过两种方式将模型表示为一个传递函数形式的线性时不变(LTI)对象：一种方法是创建两个行向量，其按降阶顺序分别包含分子分母多项式中 s 各次幂的系数；另一种方法是使用命令 tf 建立传递函数模型。tf 命令的使用格式为：

G(s) = tf(num,den)

其中，num 为传递函数的分子多项式，den 为传递函数的分母。

例 7－1 建立一控制系统的传递函数模型，系统是输入为 $x(t)$、输出为 $y(t)$的单输入单输出三阶线性定常系统。其微分方程为：

$\dddot{y} + 2.4\ddot{y} + 1.8\dot{y} + 2 = 2\ddot{x} + \dot{x} + 1$

在零初始条件下，对上式进行拉普拉斯变换，可得到系统的传递函数模型：

$$
G(s) = \frac{Y(s)}{X(s)} = \frac{2s^2 + 3s + 1}{s^3 + 2.4^2 + 1.8s + 2}
$$

建立此系统传递函数模型的 MATLAB 程序为：

```
%MATLAB program 7-1
%Construct transfer function
num=[2 3 1];                    % 输入传递函数分子多项式
den=[1 2.4 1.8 2];              % 输入分母多项式
G=tf(num,den);                  % 建立 G(s)为对象
%或者为:G=tf([2 3 1],[1 2.4 1.8 2]);
get(G);                         % 显示对象的特性
[nn,dd]=tfdata(G,'v');          %从对象中提取分子分母多项式
```

程序运行结果为：

```
nn =
```

```
        0       2       3       1
dd =
    1.00000000000000   2.40000000000000   1.80000000000000   2.00000000000000
```

将传递函数形式的分子和分母的多项式表示成一系列一阶因子连乘积的形式，可得到传递函数的另一种表示方法——零极点增益形式：

$$\frac{Y(s)}{X(s)}=G(s)=k\frac{(s-z_1)(s-z_2)\cdots(s-z_m)}{(s-P_1)(s-P_2)\cdots(s-P_n)} \tag{7-3}$$

其中，k 为增益系数，$z_i(i=1,2,\cdots,m)$为系统的零点，$P_j(j=1,2\cdots,n)$为系统的极点。

如果传递函数以零极点形式表示，可用两种方式将模型表示为一个传递函数形式的线性时不变(LTI)对象：一种方法是输入零极点列向量及标量形式的增益；另一种方法是用命令 zpk。zpk 命令的使用格式为：

```
G(s) = zpk(z,p,k)
```

其中，z 为传递函数的零点，p 为传递函数的极点，k 为传递函数的增益。

例 7-2　已知一个具有复数极点的四阶控制系统，其传递函数 $G(s)$的零点为 -2、-4，极点为 -1、$-4\pm j6$、-20，增益为 150，试建立此控制系统的传递函数模型。

建立此系统传递函数模型的 MATLAB 程序为：

```
%MATLAB program 7-2
%Construct transfer function
z=[-2;-4];                              %  输入传递函数零点
p=[-1;-4+6i;-4-6i;-20];                 %  输入传递函数极点
k=150;                                  %  输入增益
G=zpk(z,p,k);                           %  建立 G(s)为对象
%或者为:G=zpk([-2;-4],[-1;-4+6i;-4-6i;20],150)
get(G);                                 %  显示对象的特性
pzmap(G);                               %  绘制零极点图
```

程序运行结果为：

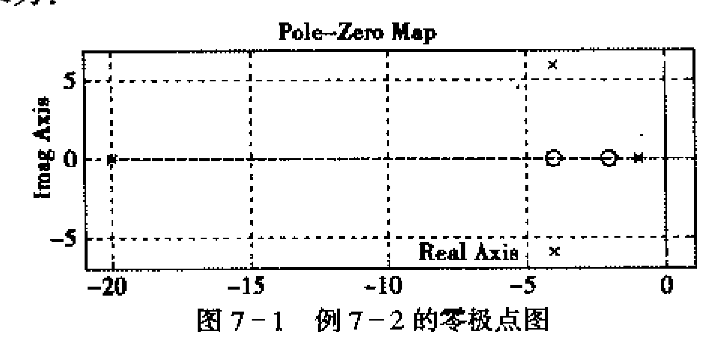


图 7-1　例 7-2 的零极点图

◆ 频率域中的数学模型——频率特性

分析和设计控制系统的另一种非常实用和重要的方法是频率响应法，频率响应法中用到的数学模型就是频率特性。频率特性可直接由传递函数得到，频率特性 $G(j\omega)$与传递函数 $G(s)$之间的关系可以表示为：

$$G(j\omega)=G(s)|_{s=j\omega} \tag{7-4}$$

由于 $G(j\omega)$是复数，可以用 $A(\omega)$来表示 $G(j\omega)$的模，称为幅频特性，用 $\psi(\omega)$来表

示 $G(j\omega)$的幅角,称为相频特性。于是,(7-4)式可写成:

$$G(j\omega)=A(\omega)e^{j\varphi(\omega)} \tag{7-5}$$

通过 MATLAB 工具箱中的函数 freqs()可绘出幅频特性和相频特性曲线。

提示:控制系统的三种输入输出数学模型之间存在着一定的内在关系,可以相互转换。

2. 状态空间模型

状态空间法是一种应用更为广泛的系统模拟分析和设计方法,它可以用于表示非线性系统、多变量系统,并且可以利用计算机方便地求出到其数值响应解。状态空间法中所要用到的数学模型就是状态空间模型。一个线性定常系统的状态空间模型为:

$$\begin{cases}\dot{x}=Ax+Bu\\ y=Cx+Du\end{cases} \tag{7-6}$$

其中,x 为状态向量,$\dot{x}$ 为状态向量对时间的微分,$\boldsymbol{y}$ 为输出向量,$\boldsymbol{u}$ 为输入或控制向量,$\boldsymbol{A}$ 为系统矩阵,$\boldsymbol{B}$ 为输入矩阵,$\boldsymbol{C}$ 为输出矩阵,$\boldsymbol{D}$ 为前向反馈矩阵。

在 MATLAB 中,函数 ss()可用来建立状态空间模型,其调用格式为:

sys = ss(**A**,**B**,**C**,**D**)

其中,$\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$ 和 $\boldsymbol{D}$ 的含义和式(7-6)中的 $\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$ 和 $\boldsymbol{D}$ 的含义相同。具体使用方法请参见例 7-3。

例 7-3 利用下面给出的矩阵在 MATLAB 中建立控制系统模型并实现不同模型之间的转换。具体要求如下:

(1) 建立控制系统的状态空间模型。

(2) 将建立的状态空间模型转换成传递函数模型。

(3) 使用不同的名称,将建立的传递函数模型转换回状态空间模型。

(4) 将原状态空间模型转换成零极点模型。

$$\boldsymbol{A}=\begin{bmatrix}-4 & 1 & 2\\ 1 & -5 & 3\\ 2 & 0 & -6\end{bmatrix},\boldsymbol{B}=\begin{bmatrix}1\\ 0.5\\ 2\end{bmatrix},\boldsymbol{C}=[2\quad 1\quad 2],\boldsymbol{D}=[0]$$

分析:该例说明了如何建立状态空间模型,以及状态空模型与传递函数模型、零极点模型之间的转换。

MATLAB 源程序如下:

```
%MATLAB program 7-3
% Construct state space model
clear all; clc;
A=[-4 1 2; 1 -5 3; 2 0 -6];                        % 输入模型数据
B=[1; 0.5; 2]; C=[2 1 2]; D=0;
GSS=ss(A,B,C,D)                                     % 建立状态空间模型
GTF=tf(GSS)                                         % 变换成传递函数形式
GSS1=ss(GTF)                                        % 变换回状态空间模型
GZPK=zpk(GSS)                                       % 变换成零极点形式
GSS2=ss(GZPK)
```

程序运行结果为：

```
 a =
        x1    x2    x3
  x1   -4     1     2
  x2    1    -5     3
  x3    2     0    -6
b =
        u1
  x1    1
  x2   0.5
  x3    2
c =
        x1   x2   x3
  y1    2    1    2
d =
        u1
  y1    0
 Continuous-time model.
 Transfer function:
  6.5 s^2 + 83 s + 266
 ---------------------------
s^3 + 15 s^2 + 69 s + 88
  a =
          x1       x2        x3
   x1     15    -4.313    -1.375
   x2    16        0          0
   x3     0        4          0
   b =
         u1
   x1    4
   x2    0
   x3    0
   c =
          x1       x2       x3
   y1   1.625    1.297   1.039
   d =
         u1
   y1    0
   Continuous-time model.
   Zero/pole/gain:
    6.5 (s^2 + 12.77s + 40.92)
  ------------------------------------
  (s+2.099) (s^2 + 12.9s + 41.92)
```

```
a =
          x1       x2       x3
   x1   -6.45      1      -0.2964
   x2   -0.31    -6.45    -0.3293
   x3     0        0      -2.099
b =
         u1
   x1    0
   x2    0
   x3   2.55
c =
         x1    x2    x3
   y1   1.13   0    2.55
d =
         u1
   y1    0
Continuous-time model.
```

从以上结果可以看出,从传递函数和零极点模型转换回状态空间模型的 A、B、C、D 矩阵和原来的不大相同,但是它们所表示的系统是一样的,这是由于控制系统的状态空间模型不具有惟一性所引起的。

3. 结构图模型

传递函数在控制系统中具有非常重要的地位,它可以通过描述系统变量之间关系的方框图(即结构图)的形式表现出来。结构图是控制系统普遍采用的表示形式。根据联接方式不同,控制系统的结构图可分为以下三种形式:

◆ **串联形式**

设控制系统由三个子系统 $G_1(s)$、$G_2(s)$、$G_3(s)$ 串联而成,则系统的传递函数 $G(s)$ 为:

$$G(s)=G_1(s)G_2(s)G_3(s) \tag{7-7}$$

其结构图如图 7-2 所示。

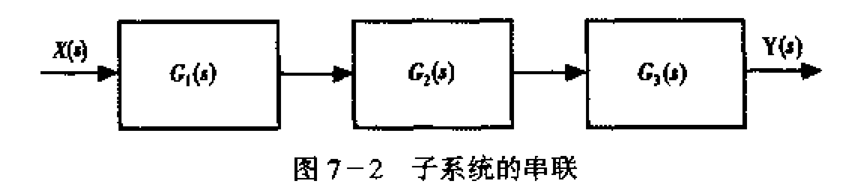


图 7-2　子系统的串联

◆ **并联形式**

设控制系统由三个子系统 $G_1(s)$、$G_2(s)$、$G_3(s)$ 并联而成,则系统的传递函数 $G(s)$ 为:

$$G(s)=G_1(s)\pm G_2(s)\pm G_3(s) \tag{7-8}$$

其结构图如图 7-3 所示。

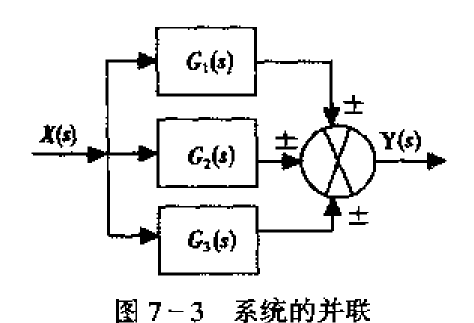


图 7－3　系统的并联

◆ **反馈形式**

设控制系统由两个子系统 $G_1(s)$、$H(s)$反馈联接而成，则系统的传递函数 $G(s)$为：

$$G(s)=\frac{G_1(s)}{1\pm G_1(s)H(s)} \tag{7-9}$$

其结构图如图 7－4 所示。

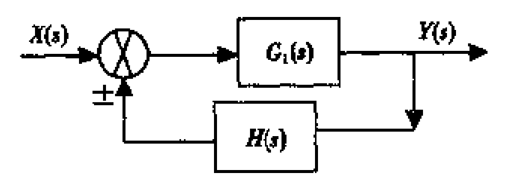


图 7－4　子系统的反馈联接

为了便于进行控制系统数学模型之间的转换，在表 7－1 中列出了 MATLAB 提供的用于模型转换的函数。

表 7－1　模型转换函数表

函　数	格　式	说　明
SS2TF	[B,A] = SS2TF(A,B,C,D,IU)	用于将状态空间模型转换成传递函数模型
SS2ZP	[Z,P,K] = SS2ZP(A,B,C,D,I)	用于将状态空间模型转换成零极点模型
TF2SS	[A,B,C,D] = TF2SS(B,A)	用于将传递函数模型转换成状态空间模型
TF2ZP	[Z,P,K] = TF2ZP(B,A)	用于将传递函数模型转换成零极点模型
ZP2TF	[Z,P,K] = TF2ZP(B,A)	用于将零极点模型转换成传递函数模型
ZP2SS	[A,B,C,D] = ZP2SS(Z,P,K)	用于将零极点模型转换成状态空间模型

7.1.2　性能指标

性能指标是评价所设计出的控制系统的性能好坏的标准，这里只对系统的动态指标进行介绍。控制系统的动态指标是在单位阶跃响应的情况下定义的，主要包括：

◆ **超调量(σ%)**

超调量是指控制系统阶跃响应曲线 $h(t)$超出稳态值的最大值与稳态值的比值，一般用百分数来表示，即：

$$\sigma\% =\frac{h(t_p)-h(\infty)}{h(\infty)}\times 100\% \tag{7-10}$$

◆ **调节时间(t_s)**

调节时间是指在控制系统阶跃响应曲线中,$h(t)$进入稳态值附近±5%$h(\infty)$(或者±2%$h(\infty)$)的误差带而不再超出的最小时间,也称为过渡过程时间。

◆ **峰值时间(t_p)**

峰值时间是指从0到阶跃响应曲线$h(t)$中超过其稳态值而达到第一个峰值之间所需要的时间。

◆ **上升时间(t_r)**

上升时间是指在控制系统阶跃响应曲线中,响应从0.1 $h(\infty)$~0.9 $h(\infty)$所需要的时间。

◆ **恢复时间**

恢复时间是指从阶跃扰动作用开始到输出量基本上恢复稳态的过程中,输出量与新稳态值之差进入某基准量的±5%(或者±2%)范围之内所需的时间。

◆ **稳态误差系数**

稳态误差系数是描述系统稳态误差特性的性能指标。针对不同的输入信号,有不同的稳态误差系数,如表7-2所示。

表7-2 稳态误差系数表

型号//输入		1(T)	T*1(T)	(1/2)T^2*1(T)
0型	误差系数	K_P=CONST	K_V=0	K_A=0
	稳态误差	1/(1+ K_P)	∞	∞
1型	误差系数	K_P=∞	K_V= CONST	K_A=0
	稳态误差	0	1/ K_V	∞
2型	误差系数	K_P=∞	K_V=∞	K_A= CONST
	稳态误差	0	0	1/K_A

对于控制系统的性能指标的具体含义,下面通过一个实例的阶跃响应曲线加以说明。

例7-4 有一个控制系统的传递函数为:

$$G(s)=\frac{10(s+3)(s+2.5)}{(s+5)(s+2)(s+15)}$$

试绘制出其阶跃响应曲线,并指出系统的各性能指标值。

MATLAB源程序为:

```
%MATLAB program 7-4
%Meanings of performance criterions of control system
clear all; clc;
z=[-3;-2.5]; p=[-5;-2;-15]; k=10;
G=zpk(z,p,k);
step(G);
```

程序运行结果如图7-5所示。

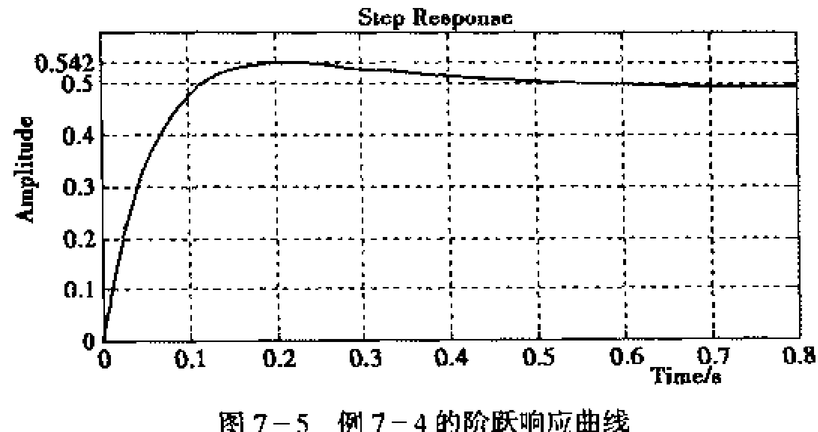


图7-5 例7-4的阶跃响应曲线

在控制系统中,最重要的性能指标是超调量、调节时间。在图7-5中,超调量为0.542,峰值时间为0.18s,调节时间为0.45s。

7.2 控制系统仿真

将MATLAB与控制系统工具箱结合进行控制系统仿真,是学习控制理论和研究自动控制系统过程中非常重要的一部分,因为在对控制系统进行建模、分析和设计时,需要非常复杂的数学运算,如实数和复数的标量、向量、矩阵等。显然,在当今的信息时代,复杂的数学计算主要依靠计算机来完成。而MATLAB是功能非常强大的科学计算和仿真软件,所以,学会使用MATLAB仿真技术来进行控制系统仿真是非常必要的。

7.2.1 仿真模块

当在MATLAB命令窗口中键入simulink命令并回车后便可进入到MATLAB的动态仿真环境,并打开仿真模块库浏览器。控制系统仿真所需要的仿真模块,大多存放在标准仿真子模块库中,只有一少部分存放在仿真模块控制系统工具箱中。下面对实现控制系统仿真时所需的仿真模块进行介绍。

1.Control System Toolbox(控制系统工具箱)

打开控制系统工具箱,便可看见其中所包含的仿真模块:Input Point(输入节点)、Output Point(输出节点)和linear time invariable(线性时不变, LTI)系统模块,如图7-6所示。

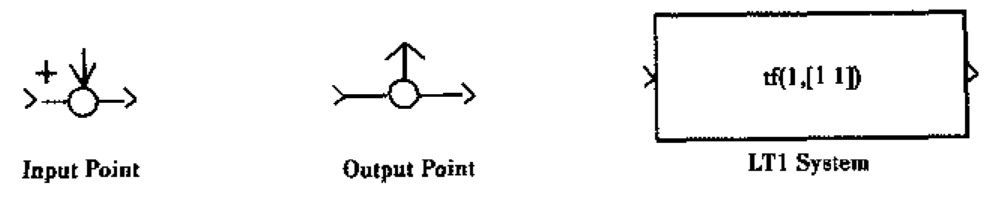


图7-6 控制系统工具箱中的仿真模块

◆ **Input Point(输入节点)**

Input Point(输入节点)主要用于源信号的输入。但在控制系统仿真中,常常不用这个节点作为输入节点,而是将它省略掉。

◆ **Output Point(输出节点)**

Output Point(输出节点)主要用于信号的输出。但在控制系统仿真中,常常不用这个节点作为输出节点,而是将它省略掉。

◆ **linear time invariable(线性时不变系统模块, LTI)**

linear time invariable(线性时不变系统, LTI)模块主要用于定义控制系统中各种各样的线性时不变对象,即线性时不变控制系统的模型,如传递函数模型、状态空间模型、零极点模型,可以用单个变量名来表示整个控制系统的模型。

构造线性系统模型可以采用 7.1 节所介绍的方法来完成,如采用下面的 MATLAB 语句:

```
G=zpk(z,p,k);
G=tf(num,den);
G=ss(A,B,C,D)
```

上面三种不同模型的创建方法可根据系统不同的数学模型来选取。接下来通过实例加以说明。

例 7-5 利用线性时不变系统模块对控制系统进行仿真,控制系统的状态方程中各矩阵为:

$$\boldsymbol{A}=\begin{bmatrix}0 & 1 & 0\\0 & 0 & 1\\-8 & -14 & -7\end{bmatrix},\boldsymbol{B}=\begin{bmatrix}0\\0\\3\end{bmatrix},\boldsymbol{C}=[1],\boldsymbol{D}=0;$$

要仿真此系统,首先需要建立一个 M 文件,并在命令窗口中运行此 M 文件。M 文件的程序代码为:

```
%MATLAB program 7-5
% Construct LTI system
A=[0 1 0;0 0 1;-8 -14 -7];
B=[0;0;3];
C=[1 0 0];
D=0;
G=SS(A,B,C,D);
```

然后在 SIMULINK 环境下创建系统仿真模型,如图 7-7 所示。

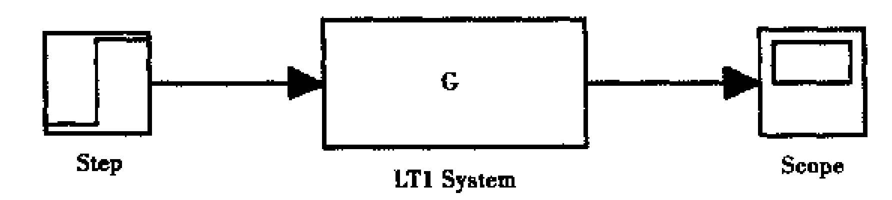


图 7-7 例 7-5 中的系统仿真模型

其中,对象 LTI System 的参数设置如图 7-8 所示。



图 7－8　LTI System 对象的参数设置

完成各项的参数设置后，运行该系统得到的仿真结果如图 7－9 所示。

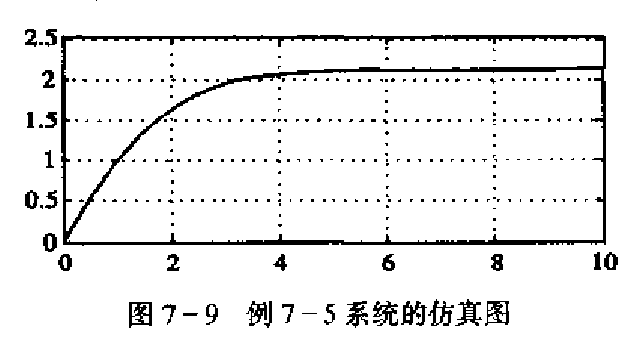


图 7－9　例 7－5 系统的仿真图

提示：本例中系统模型的创建也可通过在 MATLAB 命令窗口中将各矩阵的值和模型创建命令逐一输入完成。

2．Simulink 子模块库

在 MATLAB 命令窗口中输入命令 simulink3，或者先进入 Simulink Library Brower 窗口，再双击 Simulink 子模块库，即可打开如图 7－10 所示的模块库。

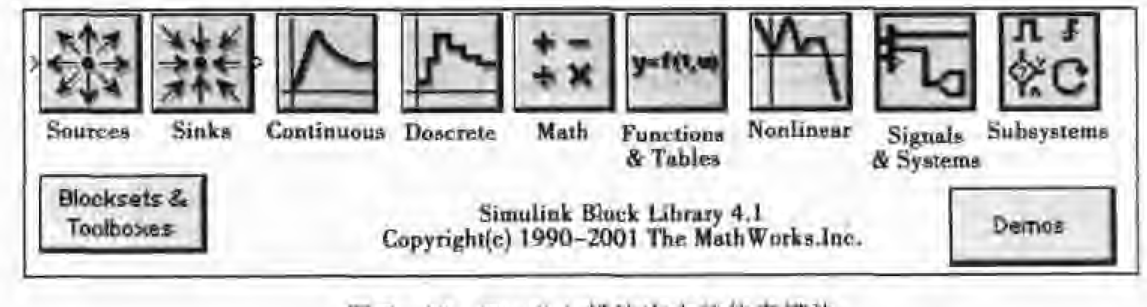


图 7－10　Simulink 模块库中的仿真模块

Simulink 子模块库中又包含：Sources（信号源子模块集）、Sinks（信号接收子模块集）、Continuous（连续子模块集）、Discrete（离散子模块集）、Math（数学运算子模块集）、Func-

tions & Tables(函数与表格子模块集)、Nonlinear(非线性子模块集)、Signals & Systems(信号与系统子模块集)、Subsystems(子系统子模块集)等 9 个子模块集。下面只介绍与控制系统仿真有关的子模块集。

◆ **Sources(信号源子模块集)**

在 Sources(信号源子模块集)包含各种各样的输入信号模块,如图 7－11 所示。

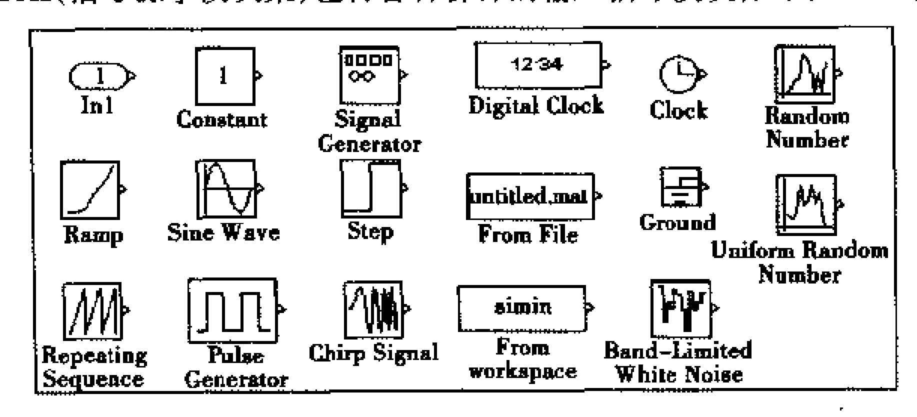


图 7－11　信号源子模块集

下面对信号源子模块集中的主要输入信号模块进行介绍:

- Signal Generator(信号发生器):它不仅能够产生若干种常用信号,如正弦波信号、方波信号、锯齿波信号和随机信号,通过它还可以调整信号的幅值和相位。
- Step(阶跃信号)模块:它能够根据不同的参数产生不同的阶跃信号,可以对采样时间、信号初始值、信号终值和产生阶跃的时间等参数进行相应地修改。
- From File(从文件获得输入信号)模块:通过此模块可以从指定的文件中获得任意的输入信号。
- From Workspace(从工作空间获得输入信号)模块:通过此模块可以从当前的工作空间中获得任意的输入信号。

其他类型的信号输入模块,由于在控制系统仿真中几乎不用,故此处不作详细介绍。

◆ **Sinks(信号接收子模块集)**

Sinks(信号接收子模块集)中包含用于显示输出信号或者显示计算结果的仿真模块,如图 7－12 所示。

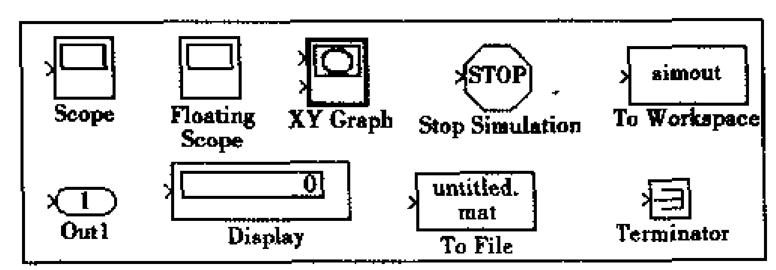


图 7－12　信号接收子模块集

下面对 Sinks(信号接收子模块集)中的主要输出信号模块进行介绍:

- Scope(示波器):用于将输入信号在示波器上显示出来。
- Floating Scope(浮动示波器):此模块可以在仿真模型中单击某一信号线,而显示

出该信号线上的信号。

- XY Graph(X-Y坐标图):用于将两路输入信号分别作为XY坐标图的两个坐标轴的值,并绘出它们的曲线图。
- To Workspace(输出到工作空间)模块:用于将输入信号直接输出到MATLAB的工作空间中,该模块默认的数据类型是结构体型的数据,通过设置可以将其改为矩阵型的。
- To File(输出到文件)模块:用于将输入信号直接输出到某一文件中。
- Stop Simulation(仿真中断)模块:如果输入的信号为非零时,将强行中断正在进行的仿真。

其他仿真信号接收模块,由于在控制系统仿真中几乎不用,故此处不作详细介绍。

◆ **Continuous(连续子模块集)**

Continuous(连续子模块集)中包含若干子模块集,如图7-13所示。下面对其中的主要模块进行介绍:

- Integrator(积分器):用于将输入信号进行数值积分后再输出,这是一个很常用的仿真模块。
- Derivative(微分器):用于将输入信号进行微分后再输出,一般应尽量避免使用它。
- State-Space(状态空间)模块:这是线性系统的一种时域描述,当采用状态空间法分析和设计控制系统时,就得使用该模块表示状态方程。
- Transfer Fcn(传递函数)模块:这是在控制系统分析和设计中经常用到的模块,控制系统的数学模型常常表示成传递函数形式。
- Pole-Zero(零极点)模块:当系统的传递函数用零极点形式表示时,通过该模块可直接输入控制对象的参数。

其他连续仿真模块,由于在控制系统仿真中几乎不用,故此处不作详细介绍。

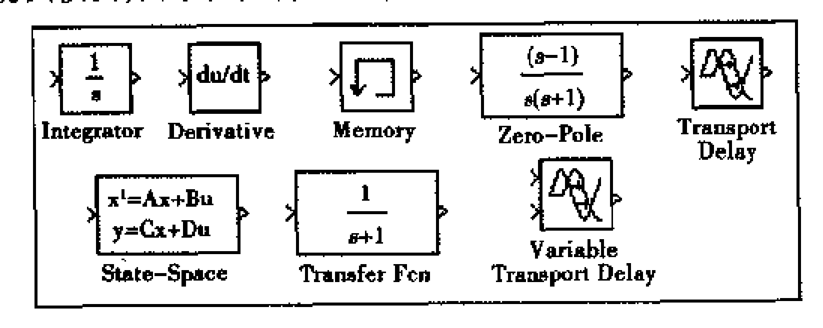


图7-13 连续子模块集

◆ **Math(数学运算子模块集)**

Math(数学运算子模块集)中主要包含Sum(求和)、Product(乘积)、Gain(数值增益)、Matrix Gain(矩阵增益)、Logical Operator(逻辑运算)、Math Function(常用的数学函数)、Abs(求绝对值或求模)和Sign(符号函数)等各种数学运算仿真模块,如图7-14所示。

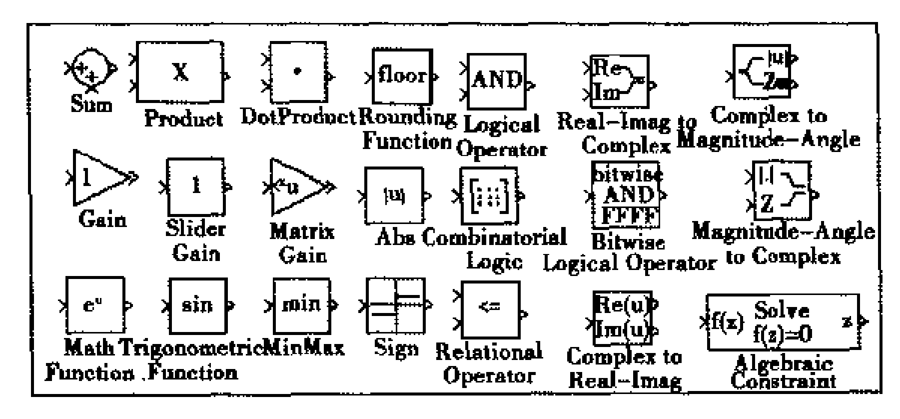


图 7-14　数学运算子模块集

7.2.2　仿真模型

一个控制系统通常由多个元部件相互连接而成，其中每个元部件都可以用一组微分方程或传递函数来表示。控制系统仿真模型的建立主要与各子系统的仿真模型的联接方式有关。下面主要讨论常用的三种基本互联模型：串联、并联和反馈联接。

1. 串联系统

若串联系统的两个子系统的传递函数分别为 $G_1(s)$、$G_2(s)$，则系统总的传递函数为 $G(s)=G_1(s)G_2(s)$。由此可见，$G(s)$的零点是 $G_1(s)$和 $G_2(s)$的联合零点，$G(s)$的极点是 $G_1(s)$和 $G_2(s)$的联合极点。因此，两个子系统串联后可能会出现零极点对消问题和零极点重数增加问题。

在 MATLAB 中，实现两个子系统串联的语句为：

G=G1*G2;

下面通过实例说明在 MATLAB 中如何创建串联系统。

例 7-6　建立由两个子系统组成的串联控制系统的传递函数模型。其中，两个子系统的传递函数 $G_1(s)$、$G_2(s)$分别为：

$$G_1(s)=\frac{3s^2+2s+1}{4s^2+5s+8},\quad G_2(s)=\frac{4s}{s^2+2s+10}$$

试在 MATLAB 中实现这两个子系统的串联联接，给出该控制系统的传递函数和以零极点形式表示的数学模型，并绘出单位阶跃响应曲线图。

MATLAB 实现程序如下：

```
%MATLAB program 7-6
% Serial control system
clear all;
clc;
num1=[3 2 1];                          % G1(s)的传递函数
den1=[4 5 8];
G1=tf(num1,den1);
num2=[4 0];                            % G2(s)的传递函数
```

```
den2=[1 2 10];
G2=tf(num2,den2);
Gtf=G1*G2                    % 以传递函数形式串联
Gzpk=zpk(Gtf)                % 以零极点形式串联
step(Gtf);
hold on;
step(Gzpk);
hold off;
```

程序运行后的阶跃响应图如图7－15所示。

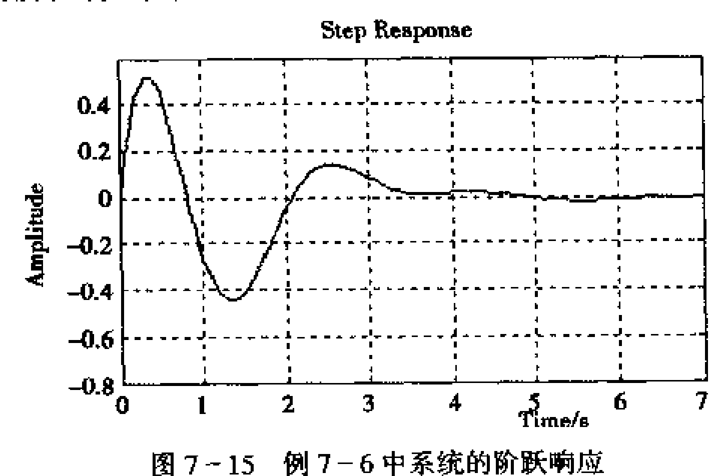


图7－15　例7－6中系统的阶跃响应

程序运行后的结果如下：

```
Transfer function:
        12 s^3 + 8 s^2 + 4 s
---------------------------------------
4 s^4 + 13 s^3 + 58 s^2 + 66 s + 80
Zero/pole/gain:
  3 s (s^2 + 0.6667s + 0.3333)
---------------------------------------
(s^2 + 1.25s + 2) (s^2 + 2s + 10)
```

2．并联系统

若并联系统的两个子系统的传递函数分别为$G_1(s)$、$G_2(s)$，则系统总的传递函数为$G(s)=G_1(s)+G_2(s)$。在MATLAB中，实现两个子系统并联的语句为：

```
G=G1+G2;
```

下面通过实例说明在MATLAB中如何创建并联系统。

例7－7　建立由两个子系统组成的并联控制系统的传递函数模型。其中，两个子系统的传递函数$G_1(s)$、$G_2(s)$分别为：

$$G_1(s)=\frac{5s+4}{2s^2+4s+12},\quad G_2(s)=\frac{3s^2+s+4}{5s^2+12s+3}$$

试在MATLAB中实现这两个子系统的并联联接，给出该控制系统的传递函数和以

零极点形式表示的数学模型，并绘出单位阶跃响应曲线图。

MATLAB实现程序如下：

```
%MATLAB program 7-7
% Parallel control system
clear all;
clc;
num1=[5 4];                      % G1(s)的传递函数
den1=[2 4 12];
G1=tf(num1,den1);
num2=[3 1 4];                    % G2(s)的传递函数
den2=[5 12 3];
G2=tf(num2,den2);
Gtf=G1+G2                        % 以传递函数形式并联
Gzpk=zpk(Gtf)                    % 以零极点形式并联
step(Gtf);
hold on;
step(Gzpk);
hold off;
```

程序运行后的结果如下：

```
Transfer function:
6 s^4 + 39 s^3 + 128 s^2 + 91 s + 60
------------------------------------
10 s^4 + 44 s^3 + 114 s^2 + 156 s + 36
Zero/pole/gain:
0.6 (s^2 + 0.7012s + 0.6) (s^2 + 5.799s + 16.67)
------------------------------------
        (s+2.117) (s+0.2835) (s^2 + 2s + 6)
```

程序运行后得到的阶跃响应图如图7-16所示。

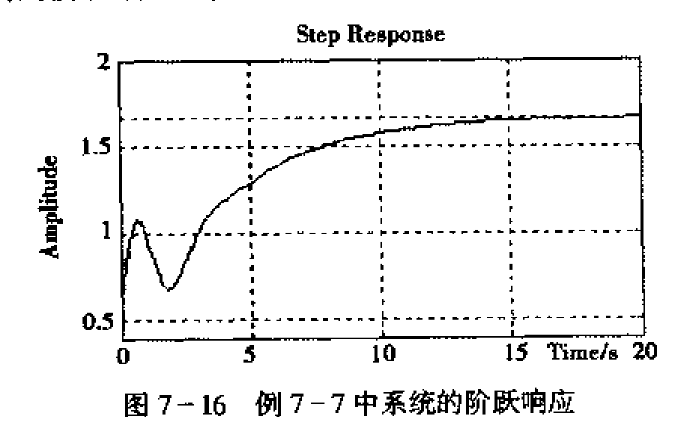


图7-16　例7-7中系统的阶跃响应

3. 反馈系统

设控制系统由两个子系统$G_1(s)$、$H(s)$反馈联接而成，如图7-4所示。一般称$G(s)$为前向传递函数，称$H(s)$为反馈传递函数。闭环传递函数的零点是$G(s)$的零点，闭环传递函

数的极点是不同于 $G(s)$ 和 $H(s)$ 的极点，因此，当开环系统不稳定时，其闭环系统可能是稳定的。反馈有两种形式：正反馈和负反馈。在控制系统设计中，常用的是负反馈。

MATLAB 控制系统工具箱提供了用于建立反馈系统传递函数的 feedback 命令，其命令调用格式为：

G= feedback(G1,H);

对于单位反馈，其命令调用格式为：

G= feedback(G1,1);

其中，第二个参数 1 表示一个具有单位增益的系统。

下面通过实例说明在 MATLAB 中如何创建反馈系统。

例 7-8　使用 MATLAB 建立一反馈控制系统，其前向通道传递函数 $G_1(s)$ 和反馈通道传递函数 $H(s)$ 分别为：

$$G_1(s)=\frac{3(s+6)}{(s+1)(s^2+3s+5)},\quad H(s)=\frac{5s+1}{15s+1}$$

试在 MATLAB 中实现这两个子系统的反馈联接，给出该控制系统的传递函数和以零极点形式表示的数学模型，并绘出单位阶跃响应曲线图。

MATLAB 实现程序如下：

```
%MATLAB program 7-8
% Feedback control system
clear all;
clc;
G1 = tf(3 * [1 6],conv([1,1],[1 3 5]));     % G1(s)的传递函数
H= tf([5 1],[15 1]);                        % H(s)的传递函数
Gtf - feedback(G1,H)                        % 以传递函数形式并联
Gzpk = zpk(Gtf)                             % 以零极点形式并联
step(Gtf);
hold on;
step(Gzpk);
hold off;
```

系统的阶跃响应曲线如图 7-17 所示。

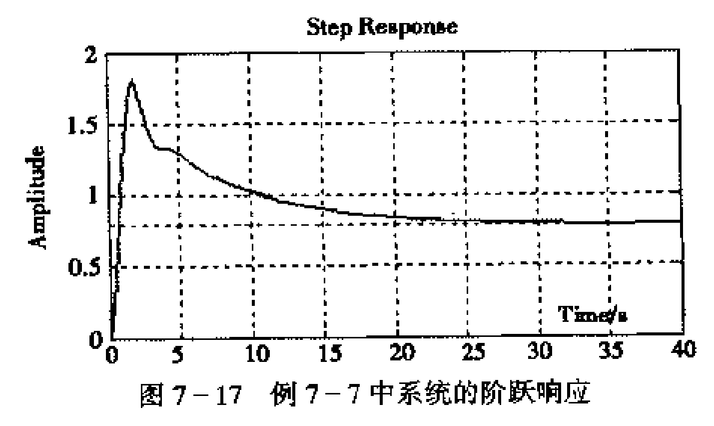


图 7-17　例 7-7 中系统的阶跃响应

程序运行后的结果如下：

Transfer function:

```
          45 s^5 + 453 s^4 + 1470 s^3 + 2481 s^2 + 1509 s + 90
------------------------------------------------------------
15 s^7 + 121 s^6 + 503 s^5 + 1295 s^4 + 2144 s^3 + 2195 s^2 + 1064 s + 115
Zero/pole/gain:
          3 (s+6) (s+1) (s+0.06667) (s^2 + 3s + 5)
------------------------------------------------------------
(s+2.142) (s+1) (s+0.1466) (s^2 + 3s + 5) (s^2 + 1.778s + 4.883)
```

提示：在实际的控制系统中，常常采用混合联接方式，既有串联、并联，还有反馈联接，这在后面的控制系统仿真实例中可见到。

7.2.3 仿真命令

MATLAB控制系统工具箱提供了大量的命令用于实现控制系统的仿真，这些命令包括模型创建命令、模型变换命令、模型简化命令、模型实现命令、模型特性命令、时域响应命令、频率域响应命令、增益选择命令、根轨迹命令、方程求解命令、演示示例命令等等。这些命令涵盖了单变量和多变量控制系统分析、设计的各个方面的计算、作图、输出打印等。下面对其中的常用命令作简要介绍。

1. 模型创建命令

模型创建命令主要用于建立控制系统的仿真模型。其中包括：

◆ **series命令**

series命令用于将两个线性时不变(LTI)子系统进行串联联接。该命令的使用格式为：

```
SYS = SERIES(SYS1,SYS2,OUTPUTS1,INPUTS2);
```

其中，SYS1、SYS2和SYS分别为子系统1、子系统2和两个子系统串联构成的系统的模型，向量OUTPUTS1为子系统1的输出变量，向量INPUTS2为子系统2的输入变量。如果参数OUTPUTS1和INPUTS2省略，则命令调用后SYS为：

```
SYS = SYS2×SYS1;
```

如果SYS1和SYS2为两个LTI子系统的矩阵，则命令格式为：

```
SYS(:,:,k) = SERIES(SYS1(:,:,k),SYS2(:,:,k),OUTPUTS1,INPUTS2);
```

调用该命令后得到的SYS矩阵的维数与SYS1和SYS2相同。

◆ **parallel命令**

parallel命令用于将两个LTI子系统进行并联联接。该命令的使用格式为：

```
SYS = PARALLEL (SYS1, SYS2, IN1, IN2, OUT1, OUT2);
```

其中，SYS1、SYS2和SYS分别为子系统1、子系统2和两个子系统并联构成的系统的模型，IN1和OUT1分别为子系统1的输入和输出，IN2和OUT2分别为子系统2的输入和输出，如果IN1、N2、OUT1和OUT2都省略，则命令调用后SYS为：

```
SYS = SYS2 + SYS1
```

如果SYS1和SYS2为两个LTI子系统的矩阵，则命令格式为：

```
SYS(:,:,k) = PARALLEL(SYS1(:,:,k),SYS2(:,:,k),IN1,...);
```

调用该命令后得到的 SYS 矩阵的维数与 SYS1 和 SYS2 相同。

◆ **feedback 命令**

feedback 命令用于将两个 LTI 子系统进行反馈联接。该命令的使用格式为:

```
SYS = FEEDBACK(SYS1,SYS2)
```

该命令格式默认是负反馈,如果是正反馈,则命令格式为:

```
SYS = FEEDBACK(SYS1,SYS2,+1)
```

如果用 SIGN 的符号表示正负反馈,则该命令的通用格式为:

```
SYS = FEEDBACK(SYS1,SYS2,FEEDIN,FEEDOUT,SIGN)
```

其中,SYS1、SYS2 和 SYS 的含义与前面命令一样,FEEDIN 和 FEEDOUT 分别为反馈通道的输入和输出。

如果 SYS1 和 SYS2 为两个 LTI 子系统的矩阵,则命令格式为:

```
SYS(:,:,k) = FEEDBACK(SYS1(:,:,k),SYS2(:,:,k))
```

调用该命令后得到的 SYS 矩阵的维数与 SYS1 和 SYS2 相同。

◆ **conv 命令**

conv 命令用于两个向量的卷积或两个多项式的乘积,调用格式为:

```
C = CONV(A, B)
```

当 conv 命令用于卷积运算时,返回向量 C 的长度为向量 A 和向量 B 的长度之和减 1;当 A 和 B 分别为两个多项式系数时,该命令相当于两个多项式的乘积。

2. 模型变换命令

常用的模型变换命令主要用于实现传递函数模型、零极点模型和状态空间模型三者之间的变换,其命令已经在表 7－1 中列出。

3. 模型特性命令

◆ **roots 命令**

roots 命令用于求多项式的根,其命令格式为:

```
ROOTS(C);
```

◆ **dcgain 命令**

dcgain 命令用于求 LTI 系统模型的增益,命令格式为:

```
K = DCGAIN(SYS)
```

如果 SYS 是维数为[NY NU S1 ... Sp] 的 LTI 模型的矩阵,调用该命令后将返回相同维数的矩阵,命令格式为:

```
K(:,:,j1,...,jp) = DCGAIN(SYS(:,:,j1,...,jp))
```

◆ **eig 命令**

eig 命令用于求方阵 X 的特征值和特征向量,命令格式为:

```
E = EIG(X)
[V,D] = EIG(X)
```

其中,E 为方阵 X 的特征值向量,D 为方阵 X 的特征值构成的对角矩阵,V 为 X 的特征向量矩阵,每一列即为一个特征向量。

◆ **ctrb 命令**

ctrb 命令用于计算可控性矩阵,命令格式为:

```
CO = CTRB(A,B)
CO = CTRB(SYS)
```

其中,CO 为矩阵 A、B 的可控性矩阵,即 CO=[B AB A^2B ...]。SYS 是用状态方程{A,B,C,D}表示的系统状态空间模型。

◆ **obsv 命令**

obsv 命令用于计算可观性矩阵,命令格式为:

```
OB = OBSV(A,C)
CO = OBSV(SYS)
```

其中,OB 为矩阵 A、B 的可观性矩阵,即 OB= [C; CA; CA^2 ...]。SYS 是用状态方程{A,B,C,D}表示的系统状态空间模型。

4. 时域响应命令

◆ **lsim 命令**

lsim 命令用于任意输入下的连续系统的时间仿真,命令格式为:

```
LSIM(SYS,U,T)
```

其中,SYS 为 LTI 系统的模型,U 为输入信号向量,T 为仿真时间向量。当 SYS 为状态空间模型且需要指定状态变量的初始值时,采用下面的命令格式:

```
LSIM(SYS,U,T,X0)
```

其中,X0 为状态变量的初始值向量。

当需要同时进行多个系统的仿真时,采用下面的命令格式:

```
LSIM(SYS1,SYS2,...,U,T,X0)
```

当需要系统输出向量 Y 时,使用下面的调用格式:

```
Y = LSIM(SYS.U,T)
```

若还需要状态变量运动轨迹及时间向量,其命令格式为:

```
[Y,T,X] = LSIM(SYS,U,T,X0)
```

◆ **step 命令**

step 命令用于绘制 LTI 系统阶跃响应曲线,其命令格式为:

```
STEP(SYS)
```

其中,SYS 可以是传递函数模型、零极点模型或状态空间模型。

当需要指定仿真时间的终值时,采用下面的命令格式:

```
STEP(SYS,TFINAL)
```

当需要按指定的时间向量进行仿真时,其命令格式为:

```
STEP(SYS,T)
```

当需要同时进行多个系统的仿真时,其命令格式为:

```
STEP(SYS1,SYS2,...,T)
```

当需要获得输出信号和时间向量时,其命令格式为:

```
[Y,T] = STEP(SYS)
```

若还需要状态变量运动轨迹及时间向量,其命令格式为:

```
[Y,T,X] = STEP(SYS)
```

5. 根轨迹命令

◆ **pzmap 命令**

pzmap 命令用于绘出单个 LTI 系统或多个 LTI 系统的零极点图，同时还可返回零点向量 Z 和极点向量 P。其命令使用格式为：

```
PZMAP(SYS)
PZMAP(SYS1,SYS2,...)
[P,Z] = PZMAP(SYS)
```

◆ **rlocus 命令**

rlocus 命令用于绘出单个 LTI 系统或多个 LTI 系统的根轨迹图，同时还可返回复数根矩阵 R 和增益 K。其命令格式为：

```
RLOCUS(SYS)
RLOCUS(SYS,K)
RLOCUS(SYS1,SYS2,...)
[R,K] = RLOCUS(SYS)
R = RLOCUS(SYS,K)
```

6. 频域响应命令

◆ **freqs 命令**

freqs 命令是拉普拉斯变换频率响应命令，在本书第 4 章中已经作过介绍。命令格式为：

```
H = FREQS(B,A,W)
[H,W] = FREQS(B,A)
```

其中，B 和 A 分别为传递函数的分于和分母多项式向量，H 和 W 分别为系统的复频响应向量和频率向量。

◆ **fbode 命令**

fbode 命令用于绘制连续时间系统的波特图，命令格式为：

```
FBODE(A,B,C,D,IU)
FBODE(NUM,DEN)
FBODE(A,B,C,D,IU,W) or FBODE(NUM,DEN,W)
[MAG,PHASE,W] = FBODE(A,B,C,D,...)
[MAG,PHASE,W] = FBODE(NUM,DEN,...)
```

其中，NUM 和 DEN 分别为系统传递函数的分子和分母多项式向量，A、B、C 和 D 为系统状态方程矩阵，W 为频率，MAG 和 PHASE 分别为幅值和相位。

7. 增益选择命令

◆ **lqr 命令**

lqr 命令用于计算连续时间系统的线性二次状态调节器设计中的最佳增益。其命令格式为：

[K,S,E] = LQR(A,B,Q,R,N)

其中,K 为最佳增益,S 为 Riccati 方程的解,E 为闭环系统的特征向量,A 和 B 为系统状态方程矩阵,Q、R 和 N 为二次型指标函数中的系数矩阵。

◆ **lqry 命令**

lqry 命令用于计算连续时间系统的线性二次输出调节器设计中的最佳增益。其命令格式为:

[K,S,E] = LQRY(SYS,Q,R,N)

其中,SYS 为系统的模型,可以是连续时间系统,也可以是离散时间系统。其余参数含义与命令 lqr 中的含义相同。

◆ **acker 命令**

acker 命令是单输入单输出系统极点配置中计算最佳增益命令。命令格式为:

K = ACKER(A,B,P)

其中,A 和 B 为状态方程矩阵,K 为增益,P 为指定的极点向量。

7.3 控制系统仿真实例

本节将通过众多实例介绍如何综合运用本章前面所介绍的知识点。所列举的实例都有很强的代表性。

例 7-9 一控制系统由 5 个子系统组成,组成结构如图 7-18 所示。

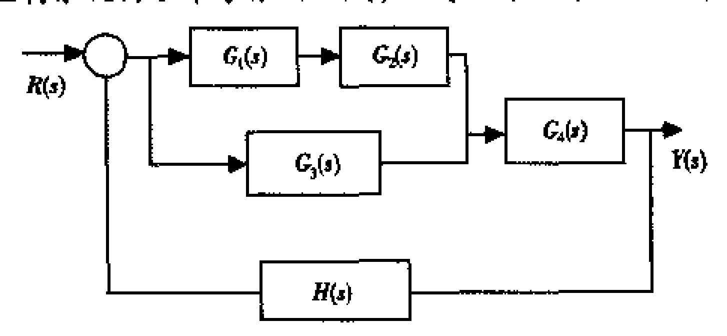


图 7-18 例 7-9 中控制系统的结构图

各子系统的传递函数分别为:

$$G_1(s)=\frac{s^2+5s+1}{2s^2+15s+6},\quad G_2(s)=\frac{4(s+6)}{(s+2)(s+20)},\quad G_3(s)=\frac{10}{s+10}$$

$$G_4(s)=\frac{s+1}{s^2+3s+6},\quad H(s)=0.1$$

试在 MATLAB 中分别用仿真模块建模和用仿真命令编程两种方法进行仿真,并绘出系统的阶跃响应曲线图。

◆ **用仿真模块建模的方法进行仿真**

首先在 Simulink 环境下将所需的仿真模块按题中要求连接起来,并将各模块的参数设置好,如图 7-19 所示。然后运行仿真系统得到阶跃响应图,如图 7-20 所示。

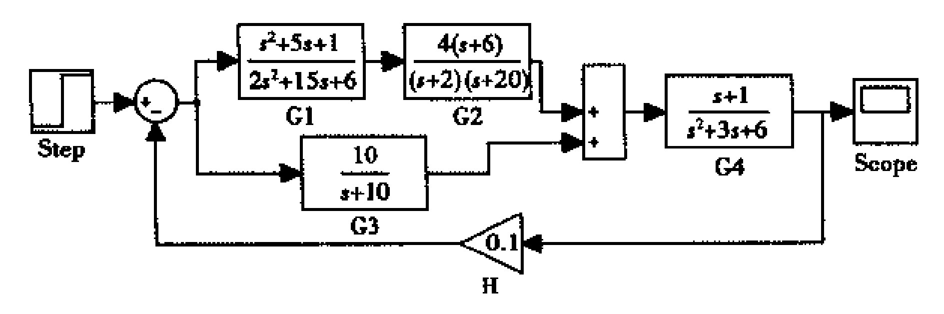


图 7－19　例 7－9 中系统的仿真模型

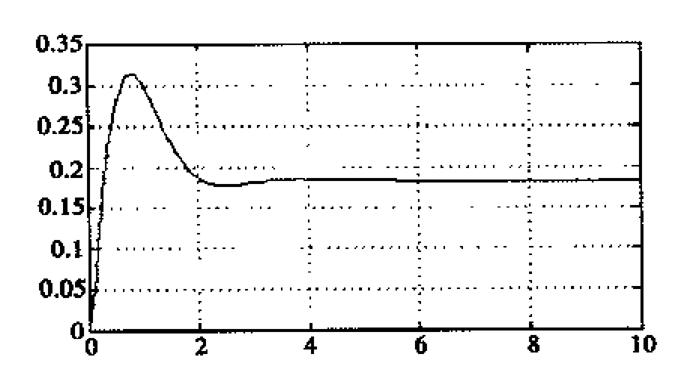


图 7－20　例 7－9 中系统的阶跃响应

◆　**用命令实现仿真**

MATLAB 程序如下：

```
%MATLAB program 7-9
%Control system simulation
clear all;
clc;
num1=[1 5 1];                          % 输入 G1
den1=[2 15 6];
G1=tf(num1,den1);
z=-6;                                  % 输入 G2
p=[-2;-20];
k=4;
G2=zpk(z,p,k);
G3=tf(10,[1 10]);                      % 输入 G3
num4=[1 1];                            % 输入 G4
den4=[1 3 6];
G4=tf(num4,den4);
H=0.1;                                 % 输入 H
Gf=(G1*G2+G3)*G4;                      % 混合联接
G=feedback(Gf,H)                       % 反馈联接
GG=tf(G)
step(G);
grid;
```

程序运行后得到的阶跃响应曲线如图 7－21 所示。

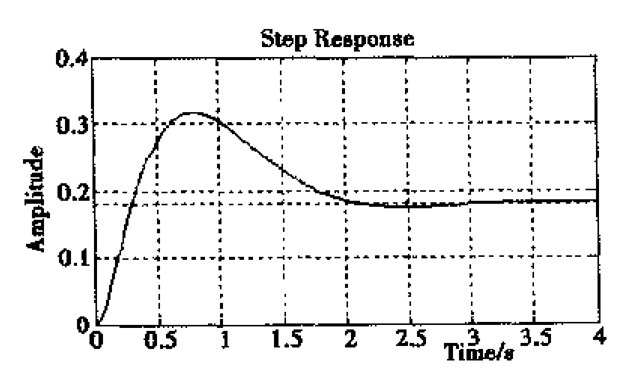


图 7－21　例 7－9 中系统的阶跃响应曲线

程序运行结果为：

```
Zero/pole/gain:
              12 (s+18.45) (s+7.054) (s+2.199) (s+1) (s+0.3845)
-----------------------------------------------------------------
(s+19.99) (s+9.88) (s+7.077) (s+1.993) (s+0.4234) (s^2 + 3.137s + 6.217)

Transfer function:
              12 s^5 + 349 s^4 + 2699 s^3 + 6654 s^2 + 5612 s + 1320
-----------------------------------------------------------------
s^7+42.5 s^6+628.7 s^5+4227 s^4+1.441e004 s^3+2.788e004 s^2+2.684e004 s+7332
```

从结果可以看出，采用仿真模块建模实现仿真的方法与采用仿真命令实现仿真的方法得到的仿真结果是完全相同的。采用哪一种方法进行仿真，可以根据实际情况而定。一般说来，采用仿真模块建模实现仿真的方法更简单、方便，但采用仿真命令实现仿真的方法更具灵活性。

例 7－10　某一单位反馈控制系统的状态方程如下：

$$\dot{x}=\begin{bmatrix}-21 & -9.25 & -3.28 & -1.56\\ 16 & 0 & 0 & 0\\ 0 & 8 & 0 & 0\\ 0 & 0 & 2 & 0\end{bmatrix}x+\begin{bmatrix}2\\0\\0\\0\end{bmatrix}u$$

$$y=[0\quad 1.56\quad 1.17\quad 1.56]x+0.5u$$

试用仿真模块建模实现仿真的方法与采用仿真命令实现仿真的方法两种方法进行仿真，并绘出系统的响应图。

1）采用仿真模块建模实现仿真

在 MATLAB 的 Simulink 环境下建立控制系统的仿真模型，如图 7－22 所示。

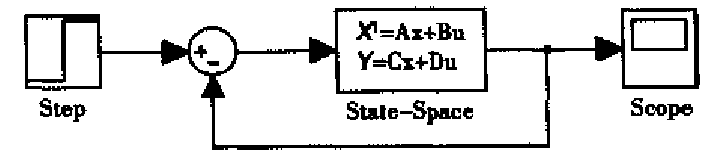


图 7－22　例 7－10 中系统的仿真模型

状态空间模型中的各参数设置如图 7－23 所示。

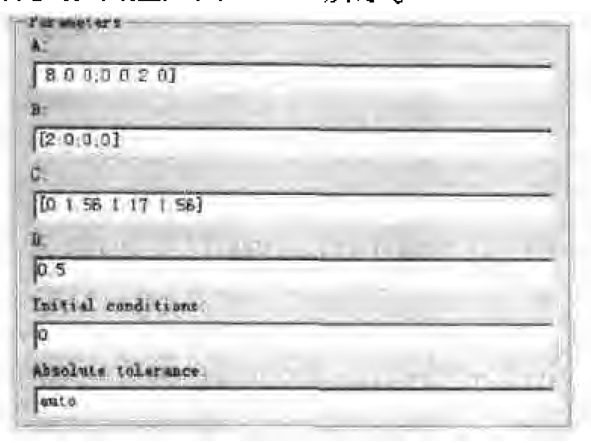


图 7－23　状态空间模型中的各参数设置

运行仿真，得到系统的阶跃响应，如图 7－24 所示。

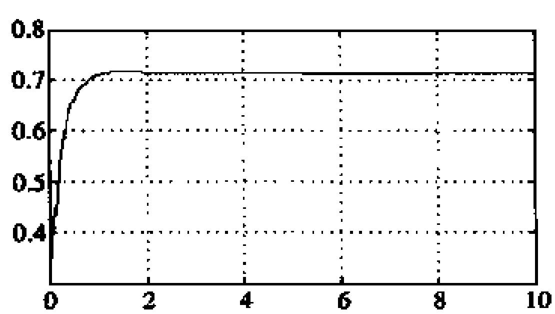


图 7－24　仿真模块建模实现仿真的阶跃响应

2) 采用仿真命令实现仿真

MATLAB 程序如下：

```
%MATLAB program 7-10
%State space mode
clear all;
clc;
A=[-21 -9.25 -3.28 -1.56;16 0 0 0;0 8 0 0;0 0 2 0];      % 输入状态方程中的各矩阵
B=[2;0;0;0];
C=[0 1.56 1.17 1.56];
D=0.5;
[num1,den1]=ss2tf(A,B,C,D);                               % 状态空间模型转换成传递函数
                                                          % 模型
G1=tf(num1,den1);
G=feedback(G1,1);                                         % 反馈联接
[num,den]=tfdata(G,'v')
[aa,bb,cc,dd]=tf2ss(num,den)                              % 传递函数模型转换成状态空间
                                                          % 模型
sys=ss(aa,bb,cc,dd)
t=0:0.01:10;                                              % 时间
```

```
u=[0,ones(1,1000)];                                    % 阶跃信号
lsim(sys,u,t);                                         % 仿真
grid;
```

程序运行结果为：

```
num =
  1.0e+002 *
                0.005000   0.105000   1.23920000   5.09440000   9.984000
den =
  1.0e+003 *
  0.00150000  0.03150000   0.27192000  0.929280000   1.3977600000
aa =
  1.0e+002 *
  -0.21000000   -1.81280000   -6.19520000   -9.31840000
  0.01000        0            0              0
  0              0.0100000    0              0
  0              0            0.0100         0
bb =
      1
      0
      0
      0
cc =
  1.0e+002 *
  0.00000000000000  0.22186666666667  1.33120000000001  3.54986666666668
dd =
  0.33333333333333
Continuous - time model.
```

得到的该反馈系统的阶跃响应图如图 7－25 所示。

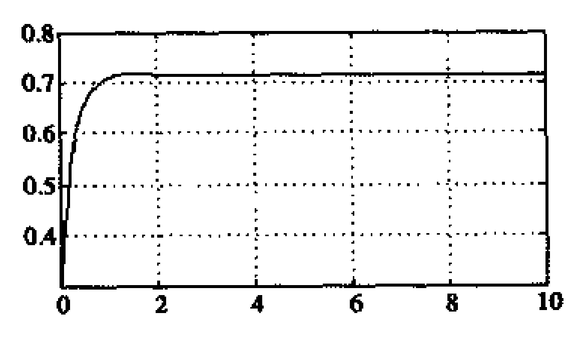


图 7－25　仿真命令实现仿真的阶跃响应

例 7－11　使用 MATLAB 绘制出以下传递函数的根轨迹：

$$G(s)=\frac{K(s+5)}{(s+1)(s+3-2i)(s+3+2i)}$$

实现绘制根轨迹的 MATLAB 程序如下：

```
%MATLAB program 7-11
```

```
%Draw the locus of roots
clear all;
clc;
num=[1 8];                                        % 输入传递函数
den=conv([1 1],conv([1 3-2*i],[1 3+2i]));
G=tf(num,den);
Gzpk=zpk(G);
rlocus(G);
axis([-20 20],[-10 10]);
```

MATLAB绘制的根轨迹如图 7-26 所示,图中显示出了各处的阻尼比的百分值。另外,单击根轨迹的某一点可计算出该点的增益、极点、阻尼比、频率等参数。例如,要计算图 7-27 中黑方框所在处的参数值,只需在那些点处单击鼠标,便可显示出所有参数的值,如图 7-27 所示。

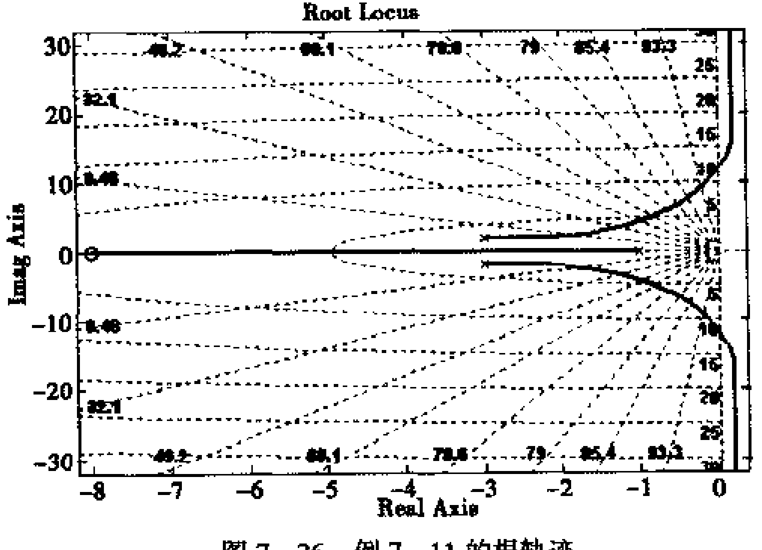


图 7-26　例 7-11 的根轨迹

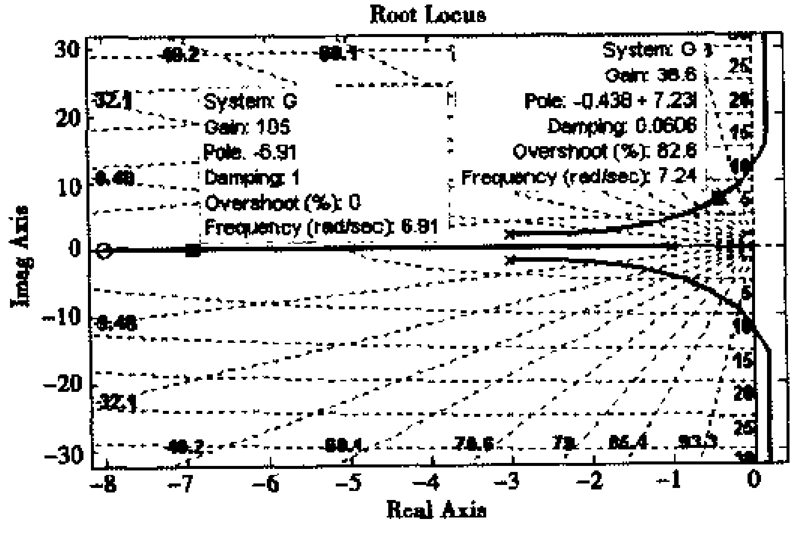


图 7-27　例 7-11 中根轨迹的参数显示

例 7-12　绘制以下传递函数的 Bode 图、Nichols 图和 Nyquist 图:

$$G(s)=\frac{40(s^2+s+32)}{(s+40)(s^2+4s+32)}$$

绘制这三种图的 MATLAB 源程序如下：

```
%MATLAB program 7-12
%Draw Bode, Nichols and Nyquist
clear all;
clc;
num=[1 1 32];
den=conv([1 40],[1 4 32]);
G=tf(40*num,den);
bode(G);                          % 绘制 Bode 图
nichols(G);                       % 绘制 Nichols 图
grid;
ngrid;
nyquist(G);                       % 绘制 Nyquist 图
axis equal;
```

运行程序绘制出的 Nichols 图、Bode 图和 Nyquist 图分别如图 7-28、图 7-29 和图 7-30 所示。

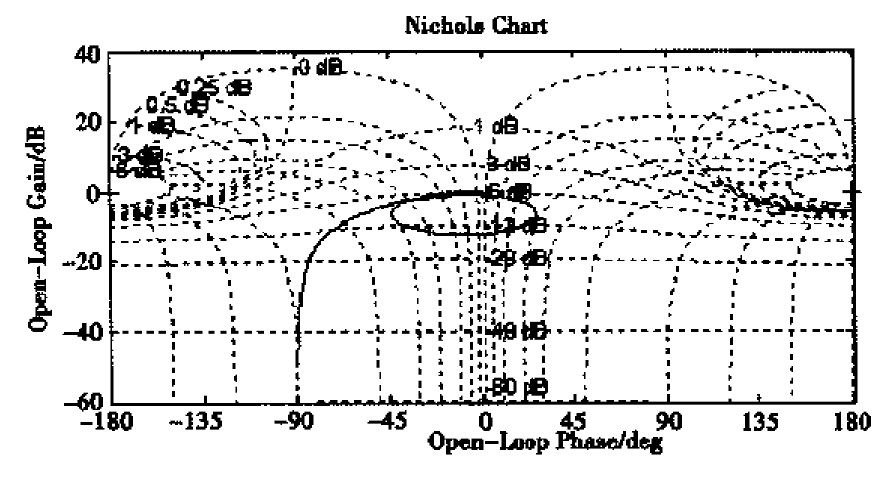


图 7-28　例 7-12 的 Nichols 图

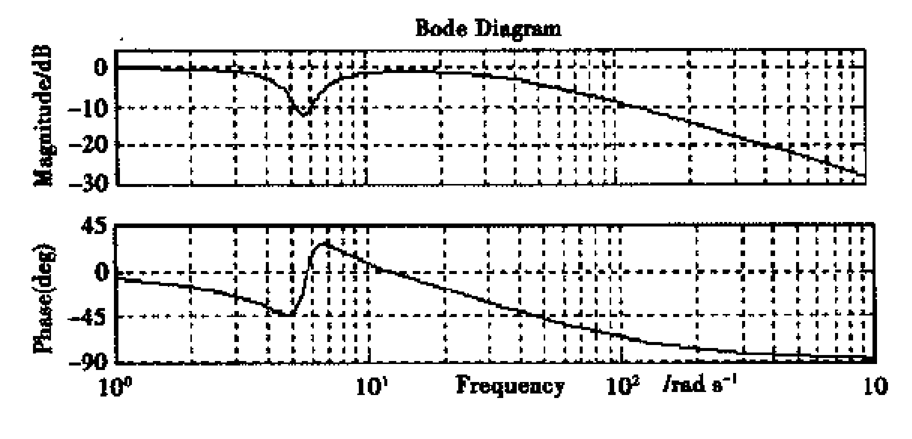


图 7-29　例 7-12 的 Bode 图

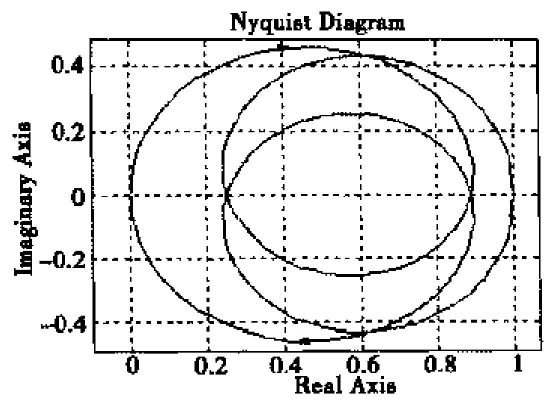


图 7-30　例 7-12 的 Nyquist 图

例 7-13　倒立摆系统的控制是控制理论应用中的一个典型范例。其中，一级倒立摆是一个单输入多输出的控制系统。其状态空间模型中的状态方程和输出方程分别为：

$$\dot{X}=\begin{bmatrix}0 & 0 & 1 & 0\\0 & 0 & 0 & 1\\0 & -0.644 & -4.26 & 0\\0 & -17.29 & 7.058 & -0.145\end{bmatrix}X+\begin{bmatrix}0\\0\\84.27\\-139.5\end{bmatrix}u$$

$$Y=\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\end{bmatrix}X$$

试设计一个二次型状态反馈调节器对其进行控制，并绘出多个输出量的阶跃响应图。

分析：倒立摆系统是一种不稳定、能控且能观系统，它的四个状态变量分别为：小车位移、小车速度、摆杆摆角和摆杆角速度，其输出为小车位移和摆杆角度变量。因此，在设计控制器对倒立摆进行控制时，应先对倒立摆系统的稳定性和能控性、能观性进行判断。这里选取二次型性能指标中的系数矩阵的值为：

$$Q=\begin{bmatrix}1 & 0 & 0 & 0\\0 & 0.001 & 0 & 0\\0 & 0 & 0.1 & 0\\0 & 0 & 0 & 0.1\end{bmatrix},R=50.0$$

MATLAB 实现程序如下：

```
%MATLAB program 7-14
%Design a controller for inverted pendulum
clear all;
clc;
A=[0 0 1 0;0 0 0 1;0 -0.644 -4.26 0;0 17.29 7.058 -0.145];        % 输入状态空间模型矩阵
B=[0;0;84.27;-139.5];
C=[1 0 0 0;0 1 0 0]; D=0;
%Computing the poles of open-loop system
E=eig(A)
%计算能控性矩阵的秩
```

```
CO=ctrb(A,B);
N_C=rank(CO)
%计算能观性矩阵的秩
OB=obsv(A,C);
N_O=rank(OB);
%计算状态反馈矩阵
q=[1,0.001,0.1,0.1];
Q=diag(q);
R=50.0;
[K p e]=lqr(A,B,Q,R);
t=0:0.02:5;                              % 仿真时间
U=ones(size(t));                         % 仿真的输入信号
[y,t]=lsim(A-B*K,B,C,D,U,t);             % 运行仿真
t=0:0.02:5;
y1=y(:,1);                               % 输出
y2=y(:,2);
plot(t,y1);
grid;
hold on;
plot(t,y2,'b-.');
程序运行后的结果为:
E =
          0
     4.0197
    -4.9548
    -3.4699
N_C =
     4
N_O =
     4
K =
    -0.1414    -0.8926    -0.2143    -0.2204
```

倒立摆系统的阶跃响应曲线如图7-31所示。

图7-31　倒立摆系统的阶跃响应

7.4　本章小结

运用 MATLAB 仿真技术，可以方便、快捷地完成各种控制系统的性能分析和控制器设计。本章详细介绍了在控制系统分析和设计中如何运用 MATLAB 这个功能强大的科学计算和仿真软件，为学习和研究控制理论提供了有益的帮助。

本章首先讨论了控制系统的数学模型(包括时域、频域中的数学模型和状态空间模型以及控制系统的结构图模型)的创建和控制系统分析和设计时必须考虑的各种性能指标；然后详细介绍了控制系统仿真所要用到的仿真模型和仿真命令；最后给出控制系统的几个仿真实例，以说明在 MATLAB 中进行控制系统仿真的具体方法与技巧。

习　题

1. 某系统由以下微分方程来描述：

$$\frac{\mathrm{d}y^3(t)}{\mathrm{d}t^3}+5\frac{\mathrm{d}y^2(t)}{\mathrm{d}t^2}+7\frac{\mathrm{d}y(t)}{\mathrm{d}t}+y(t)$$
$$=\frac{\mathrm{d}x^3(t)}{\mathrm{d}t^3}+2\frac{\mathrm{d}x^2(t)}{\mathrm{d}t^2}+3\frac{\mathrm{d}x(t)}{\mathrm{d}t}+7x(t)$$

试求该系统的传递函数$\frac{Y(s)}{X(s)}$，并转换成零极点形式表示。

2. 在 MATLAB 中，用语句给出下列系统的传递函数，并绘制系统的零极点图和阶跃响应曲线图。

(1) $G(s)=\frac{5(s+15)(s+26)}{s(s+55)(s+89)(s+47)}$

(2) $G(s)=\frac{s^4+3s^3+2s^2s+1}{s^5+4s^4+3s^3+2s^2+3s+1}$

3. 在 MATLAB 中将以下状态空间模型转换成传递函数模型，再将传递函数模型转换回状态空间模型，并绘制系统的阶跃响应曲线。

(1) $\dot{\boldsymbol{x}}=\begin{bmatrix}-4 & 1 & 0\\ 0 & -5 & 1\\ 0 & 0 & -2\end{bmatrix}\boldsymbol{x}+\begin{bmatrix}0\\0\\1\end{bmatrix}\boldsymbol{u}$

$\boldsymbol{y}=[1\quad 1\quad 0]\boldsymbol{x}$

(2) $\dot{\boldsymbol{x}}=\begin{bmatrix}-1.25 & -1 & -2 & -0.25\\ 2 & 0 & 0 & 0\\ -0.875 & -1.25 & -5 & -2.875\\ 0 & 0 & 1 & 0\end{bmatrix}\boldsymbol{x}+\begin{bmatrix}1\\0\\1.5\end{bmatrix}\boldsymbol{u}$

$\boldsymbol{y}=[-0.4375\quad -0.625\quad -1.5\quad -0.1875]\boldsymbol{x}+0.75\boldsymbol{u}$

4. 目前的计算机都要使用大型的磁盘存储装置,磁头需要在旋转磁盘的不同位置上移动且需要快速、准确地响应。假设有一个磁盘存储器的磁头定位系统的结构图如图7－32所示:

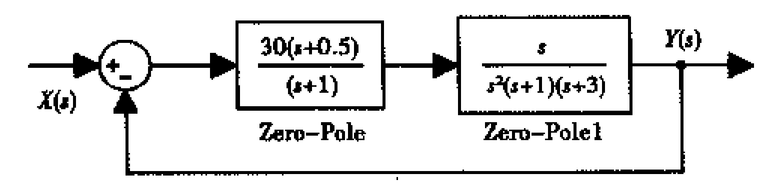


图 7－32　磁头定位系统结构图

试用时间流和数据流两种方法实现该系统的仿真。

5. 绘制图 7－33 中各系统的 Bode 图、Nichols 图和 Nyquist 图:

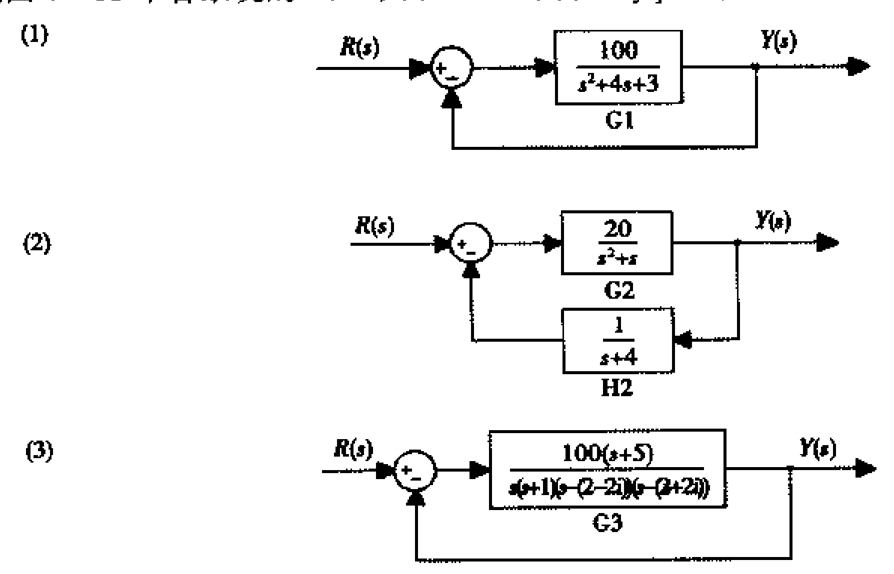


图 7－33　习题 5 中的控制系统结构图

6. 设被控系统为:

$$\dot{x}=\begin{bmatrix}-26 & -104 & 320\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}x+\begin{bmatrix}1\\0\\0\end{bmatrix}u$$

$$y=[0\quad 100\quad 500]x$$

试在 MATLAB 中为该系统设计一控制器,使其稳定。

第8章　MATLAB 在优化技术仿真中的应用

知识点：

- 优化问题与优化方法
- 遗传算法的实现及其应用
- 基于仿真的滤波器优化设计
- 基于仿真的控制系统优化设计

本章将主要讨论 MATLAB 仿真技术在优化领域中的应用。首先介绍如何利用 MATLAB 优化工具箱提供的函数求解线性规划和非线性规划问题，然后简要介绍遗传算法及程序设计，最后探讨 MATLAB 仿真技术在数字滤波器优化设计和控制系统优化设计中的应用。

通过本章的学习，读者不仅可以学习到如何利用 MATLAB 优化工具箱来求解传统优化问题，而且还可以了解到 MATLAB 在数字滤波器和控制系统优化设计方面的应用。

随着计算机科学技术的迅猛发展，计算机仿真技术也得到了长足发展，这使得许多通过传统方法难以实现的优化算法，可以很容易地实现。有许多优化算法中，如线性规划、非线性规划、遗传算法等，归根结底都是一种迭代算法，单靠人工进行迭代计算的方法显然难以胜任，而将它交给计算机去完成，既省时省力又方便快捷。

MATLAB 语言不仅在一些复杂计算（如矩阵、数组、向量、复数等）和绘图分析方面具有十分强大的功能，而且具有简单、易学易懂等优点，这使得它在优化技术领域倍受关注，得到了广泛的应用。

8.1　优化问题与优化方法

优化问题即通常所说的最优化问题，在实际的优化问题中，最优解总是难以获得，或在某些情况下，最优解根本就不存在，因此本书中将它称为优化问题，似乎更符合实际一些。

8.1.1　优化问题数学描述

优化问题是指在给定指标和元件、参数的允许取值范围条件下，确定一组独立的设计参数，使系统达到最佳技术经济性能。系统性能的优劣通常用一个关于设计参数的函数来描述。该函数称为“目标函数”，待定的设计参数称为“优化变量”，而参数范围和未包含在目标函数中的一些设计指标称之为构成优化变量的“约束条件”。寻求系统的最佳性

能，在数学上通常指的就是最小化或最大化目标函数。例如最小误差、最低成本、最大增益、最高利润等。因此，优化问题的数学模型可描述为：

$$\begin{aligned} &\min_{X\in R^n} f\ (X) \\ subject\quad to\quad & C_i(X)=0, \qquad i=1,2,\cdots,m \\ & C_i(X)\leq 0, \qquad i=m+1,\cdots,m+p \end{aligned} \tag{8-1}$$

其中，$X=[x_1,x_2,\cdots,x_n]$为优化问题的优化变量，$f(X)$为目标函数，$C_i(X)$为约束函数，并以 X^* 和 $f(X^*)$表示优化问题的最优解和最优目标函数值。

由 $\max f(X)\Leftrightarrow\min[-f((x))]$，$C_i(X)\geq 0\Leftrightarrow -C_i(X)\leq 0$ 不难看出，(8-1)式是适用于任何优化形式、任何约束条件的优化问题的一般数学描述。此外，还应该指出，上述分析不仅适用于最优化工程设计，也适用于应用科学、数学、医学、经济学、统计学等所有领域。这些领域中的数据分析、优选、决策等都可以归结为不同形式的最优化问题，并由其相应的目标函数和约束条件来描述。

8.1.2 线性规划问题

线性规划(LP)问题的一般形式可描述为：

$$(LP)\quad \begin{cases} \min(\max) Z=C^TX \\ s.t.\ a_i^TX\leq b_i \quad i=1,2,\cdots,m \\ \qquad a_i^TX\geq b_i \quad i=m+1,\cdots,l \end{cases} \tag{8-2}$$

在式(8-2)的描述中，目标函数的等值面($C^TX=$常数)为一组平行的超平面，约束条件规定了优化变量 X 的可行域。

在 MATLAB 中，新版本的最优化工具箱提供了求解线性规划问题的函数 linprog()，该函数的调用格式有以下几种：

```
X = linprog (f,A,b,Aeq,beq);
X = linprog (f,A,b,Aeq,beq,lb,ub);
X = linprog (f,A,b,Aeq,beq,lb,ub,x0);
X= linprog (f,A,b,Aeq,beq,lb,ub,x0,options);
[X,fval] = linprog (...);
[X, fval, exitflag] = linprog(...);
[X, fval, exitflag, output] = linprog(...);
[X,fval,exitflag,output,lambda] = linprog(...);
```

其中，X 为线性规划问题的解向量，f 为(8-2)式中的 C，A={a_i}，b={b_i}，Aeq 和 beq 为等式约束中的系数矩阵和向量，lb 和 ub 分别为向量 X 的下界和上界，x0 为初始搜索点。最优化运算后，结果将保存到变量 X 中，最优化的目标函数值将保存到 fval 变量中。下面通过实例说明如何求得线性规划的解。

例 8-1 求解下面的三元线性规划问题：

$$f(x)=-5x_1-4x_2-6x_3$$

$$\text{s. t.}\quad \begin{array}{l} 3x_1+2x_2+4x_3\leq 42 \\ 3x_1+2x_2\leq 20 \\ x_1,x_2,x_3\geq 0 \end{array}$$

求解此线性规划问题的 MATLAB 程序为：

```
% MATLAB program 8-1
% Solve linear program problem
clear all;
clc;
f=[-5;-4;-6];
A=[1 -1 1
  3 2 4
  3 2 0];
b=[20; 42; 30];
lb=zeros(3,1);
[x,fval] = linprog(f,A,b,[],[],lb);
```

程序运行后的结果为：

```
Optimization terminated successfully.
x =
      0.0000
     15.0000
      3.0000
fval =
    -78.0000
```

8.1.3 非线性规划问题

1. 二次规划问题

一般二次规划的数学模型为：

$$(QP)\quad \begin{cases} \min f(X)=(1/2)X^TGX+g^TX \\ s.t.\quad \mathbf{A}_1X\leq b_1 \\ \qquad\quad \mathbf{A}_2X=b_2 \end{cases} \tag{8-3}$$

其目标函数为二次型函数，G 为对称的 Hessian 矩阵，约束函数为线性函数。其中，$\mathbf{A}_1$ 为 m×n 矩阵，$\mathbf{A}_2$ 为 $m_2\times n$ 矩阵，b_1、b_2 分别为 m_1，m_2 维向量，$\mathbf{A}_1$、$\mathbf{A}_2$ 的行向量即为各个约束函数的梯度向量。

在 MATLAB 中，新版本的最优化工具箱提供了求解线性规划问题的函数 quadprog()。该函数的调用格式为：

```
x = quadprog(H,f,A,b)
x = quadprog(H,f,A,b,Aeq,beq)
x = quadprog(H,f,A,b,Aeq,beq,lb,ub)
```

```
x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0)
x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)
x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options,p1,p2,...)
[x,fval] = quadprog(...)
[x,fval,exitflag] = quadprog(...)
[x,fval,exitflag,output] = quadprog(...)
[x,fval,exitflag,output,lambda] = quadprog(...)
```

其中,H、A 和 Aeq 为矩阵,f、b、beq、lb、ub 和 x 为向量,H 为(8-3)式中的 G,f 为 g,其余参数的含义可参见线性规划问题函数中有关该参数的解释。

下面通过实例说明二次规划的求解问题。

例 8-2 求解下面的二次规划问题:

$$f(x)=0.5x_1^2+x_2^2-x_1x_2-2x_1-6x_2$$

$$\text{S. t.}\quad \begin{aligned} &x_1+x_2\leq 2\\ &-x_1+x_2\leq 2\\ &2x_2+x_2\leq 3\\ &x_1,x_2\geq 0 \end{aligned}$$

此二次规划问题可以表示成以下矩阵形式:

$$f(x)=0.5x^THx+f^Tx$$

其中,$H=\begin{bmatrix}1 & -1\\ -1 & 2\end{bmatrix}$,$f=\begin{bmatrix}-2\\ -6\end{bmatrix}$,$x=\begin{bmatrix}x_1\\ x_2\end{bmatrix}$。

求解此二次规划问题的 MATLAB 程序为:

```
% MATLAB program 8-2
% Solve quadratic programming problem
clear all;
clc;
H = [1 -1; -1 2];
f = [-2; -6];
A = [1 1; -1 2; 2 1];
b = [2; 2; 3];
lb = zeros(2,1);
[x,fval] = quadprog(H,f,A,b,[],[],lb)
```

程序运行后的结果为:

```
Optimization terminated successfully.
x =
     0.6667
     1.3333
fval =
     -8.2222
```

2. 一般非线性规划问题

有约束条件的非线性规划问题的一般描述为：

$$\min_{s.t.\,G(x)\leq 0} F(x) \tag{8-4}$$

其中，x 为一向量，$F(x)$为目标函数值，$G(x)\leqslant 0$ 为约束条件，可以是等式约束，也可以是不等式约束。

在 MATLAB 中，新版本的最优化工具箱提供了求解非线性规划问题的函数 fmincon ()。该函数的调用格式为：

```
x = fmincon(fun,x0,A,b)
x = fmincon(fun,x0,A,b,Aeq,beq)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options,P1,P2, ...)
[x,fval] = fmincon(...)
[x,fval,exitflag] = fmincon(...)
[x,fval,exitflag,output] = fmincon(...)
[x,fval,exitflag,output,lambda] = fmincon(...)
[x,fval,exitflag,output,lambda,grad] = fmincon(...)
[x,fval,exitflag,output,lambda,grad,hessian] = fmincon(...)
```

其中，各参数的含义可参见求解线性规划问题的函数中有关该参数的解释。

下面通过实例来说明有约束条件的非线性规划问题的求解问题。

例 8－3　求解以下有约束条件的非线性规划问题：

$\min f(x) = -x_1x_2x_3$

S. t.　$0\leq x_1+2x_2+2x_3\leq 72$

$x_0=[10;10;10]$

用 MATLAB 求解此有约束条件的非线性规划问题时，应先将目标函数编写成一个 M 类型的函数文件：

```
function f = myfun(x)
f = -x(1) * x(2) * x(3);
```

然后将上面的约束条件重写为下列形式：

$$-x_1-2x_2-2x_3\leq 0$$
$$x_1+2x_2+2x_3\leq 72$$

因为两个约束都是线性约束，所以可将它们写成矩阵形式：

$\boldsymbol{A}x\leqslant b$

其中，$\boldsymbol{A}=\begin{bmatrix}-1 & -2 & -2\\ 1 & 2 & 2\end{bmatrix}$，$b=\begin{bmatrix}0\\ 72\end{bmatrix}$。

求解此有约束条件的非线性规划问题的 MATLAB 程序为：

```
% MATLAB program 8-3
% Solve constrained nonlinear multivariable function
clear all;
clc;
x0 = [10; 10; 10];
A=[-1,-2,-2;1,2,2];
b=[0;72];
[x,fval] = fmincon(@myfun,x0,A,b)
```

程序运行后的结果为：

```
Optimization terminated successfully:
 Magnitude of directional derivative in search direction
  less than 2 * options.TolFun and maximum constraint violation
 is less than options.TolCon
Active Constraints:
    2
x =
   24.0000
   12.0000
   12.0000
fval =
   -3456
```

8.2 遗传算法的实现及其应用

8.2.1 遗传算法简介

遗传算法(Genetic Algorithm, GA)是模拟生物进化过程中优胜劣汰规则与群体内部染色体信息交换机制的一类处理复杂优化问题的新方法。由于 GA 不受问题本身的性质、优化准则形式、模型结构形式、被优化参数数目和有无约束等的限制，仅用目标函数在概率准则引导下进行并行全局自适应自动搜索，通过它能够处理传统优化方法难以解决的复杂问题，具有极强鲁棒性和广泛适用性，因而 GA 在各个优化领域得到了广泛应用并成为跨学科研究的热点。

标准遗传算法(Standard Genetic Algorithm, SGA)的操作过程为：

(1) 构造满足约束条件的染色体。GA 不是直接求解空间中的解，而是通过编码将解表示成适当的染色体方式来求解。GA 的编码方式有很多种，但编码方式的选取应尽可能与优化问题相符合，否则会影响计算效率。

(2)随机产生初始种群。初始种群是搜索开始的一组染色体，其数量应适当选择。

(3)计算每个染色体的适应度函数值。适应度函数是反映染色体优劣的惟一标准，GA 就是寻求适应度最大的染色体。

(4)使用复制、交叉、变异算子产生子代种群。这三个算子是 GA 的基本算子，其复制

体现了优胜劣汰的生物进化规律，交叉体现了有性繁殖的思想，变异体现了进化过程中的基因突变。

(5)重复步骤(3)、(4)，直到满足算法终止条件为止。

同传统优化算法相比，GA 具有以下 4 个突出特点：

- GA 是对参数的代码集起作用，而不是对参数本身起作用；
- GA 是从初始种群开始搜索，而不是从单点开始搜索；
- GA 在搜索过程中只使用适应度函数信息，而不用导数等信息；
- GA 使用概率转换规则而不用确定性规则。

由于这些特点，GA 的优越性主要表现在：

- 具有很强的搜索能力，找到问题的全局最优解具有更大的概率。
- 由于其固有的并行性，能有效地处理大规模的复杂优化问题。

一般而言，GA 主要由 4 个部分组成：即编码机制、控制参数、适应度函数和遗传算子。

1. 编码机制(Encoding Mechanism)

编码机制(Encoding Mechanism)是 GA 的基础，GA 不是对研究对象直接进行讨论，而是通过某种编码机制把对象统一于特定符号按一定顺序排成串。在 SGA 中，字符集由 0 和 1 组成，码为二元串。在优化问题中，一个串对应于一个可能解。在分类问题中，串可解释为一个规则，即串的前半部为输入或前件，后半部为输出或后件、结论，等等。

2. 适应度函数(Fitness Function)

在 GA 中，用适应度函数描述每一个体的适应程度。优胜劣汰是生物自然进化的法则。对于优化问题，适应度函数就是目标函数。引进适应度函数的目的在于可根据其适应度函数值对个体进行评价、比较。

3. 遗传算子(Genetic Operator)

在遗传算子中，最重要的算子有三种：选择算子(Selection Operator)、交叉算子(Crossover Operator)和变异算子(Mutation Operator)。选择算子也称复制算子(Reproduction Operator)，它的作用在于根据个体的优劣程度决定它在下一代中被淘汰还是被复制。一般说来，通过选择，将使适应度大的个体(即优良个体)有较大的生存机会，而使适应度小的个体(即低劣个体)继续存在的机会很小。SGA 采用按比例选择方式。

如果只有选择算子，GA 不会有什么新意。因为后代的群体不会超出初始群体，所以还需要一些算子来更新种群中的个体，这就是交叉和变异。交叉算子有多种形式，SGA 采用的是单点交叉方式，即从群体中随机取出两个字符串，设串长为 L，随机确定交叉点，它在 1～L－1 之间取值。然后将两个字符串的后半段互换再重新连接得到两个新串。当然，得到的新串不一定都能保留到下一代，需和原来的串进行比较，适应度大的两个将保留。进行交换后，再进行突变。变异算子是改变字符串某个位置上的字符。在 SGA 中，即为 0 与 1 的互换：0 变异为 1，1 变异为 0。

4. 控制参数(Control Parameter)

在 GA 的实际操作中,需要适当确定某些参数的值以提高优选的效果。这些参数是:字符串所含字符个数,即串长 L,SGA 中的串长为一常数;每一代群体的大小,即所包含字符串的个数,也称为群体容量 n;交换率 Pc,即施行交换操作的概率;突变率 Pm,即施行变异算子的概率。在 SGA 中,若群体容量较大,如 n=100,通常取 Pc=0.6,Pm=0.001;若群体容量较小,如 n=30,通常取 Pc=0.9,Pm=0.01。此外,还有遗传的代数、确定算法终止的指标等。

8.2.2 遗传算法程序设计

根据前面对遗传算法的介绍,可得出 GA 的算法步骤如下:

(1) 确定算法参数的初始值,包括群体容量 P、串长 L、交换率 Pc、变异率 Pm 等。

(2) 随机产生初始种群。

(3) 计算每个染色体的适应度函数值。

(4) 使用选择、交叉和变异算子产生下一代个体。

(5) 重复步骤(3)和(4),直到结束。

GA 的执行算法可描述为:

```
Begin
    t=0;
    Initialization P(t); P(t)={p1,p2,…,pn}
    Computing fitness;
    While (termination condition=True) do
          Begin
          Reproduction operation;
          Crossover operation;
          Mutation operation;
          End
    End
End
```

下面以求解一个简单优化问题 $f(x)=x_1^2+x_2^2$, ($x_1,x_2\in[-5,5]$)为例介绍遗传算法的源程序。

遗传算法的源代码如下:

```
% Genetic algorithm for solving simple optimization problem
% f(x)=x1^2+x2^2, -5≤x1≤5, -5≤x2≤5
%The stategy of optimal maintaince is applied in the algorithm

clear all;
clc;
format long;
Population_Size=100;
```

```
String_Length=30;
chromosome=round(rand(Population_Size,String_Length));
chromosome_change=zeros(Population_Size,String_Length);
flag=1;
fitness_function=zeros(1,Population_Size);
selected=zeros(1,Population_Size);
generation=1;
maxsat=100;

while(flag>0)&(generation<1000)
    sum_fit=0;
    for i=1:Population_Size
        a1=chromosome(i,1:15);a2=chromosome(i,16:30);
        param=zeros(1,2);
        for j=1:15
            param(1,1)=param(1,1)+a1(1,j)*pow2(j-1);
        end
        for j=1:15
            param(1,2)=param(1,2)+a2(1,j)*pow2(j-1);
        end
        m=pow2(15)-1;
        x1=-5+param(1,1)/m*10;
        x2=-5+param(1,2)/m*10;
        fitness_function(1,i)=x1^2+x2^2;
        if(fitness_function(1,i)<maxsat)
            maxsat=fitness_function(1,i);
            optimal=[x1 x2 fitness_function(1,i)];
        end
        if(fitness_function(1,i)<=0.001)
            flag=-1;
            optimal
            generation
            break;
        else
            sum_fit=sum_fit+fitness_function(1,i);
        end
        if (flag<0)
            break;
        end
    end


    if(flag>0)
```

```
%the first select
sum_fit=sum_fit-fitness_function(1,Population_Size);
for i=1:Population_Size-1
        x=round(rand(1)*32767);
        sum=round(sum_fit);
        rr=rem(x,sum);
        n=1;ba=1;
        partsum=0;
        while((partsum<rr)&(n<Population_Size-1))
                partsum=partsum+fitness_function(1,n);
                ba=n;
                n=n+1;
        end
        selected(1,i)=ba;
end
%reproduce
for i=1:Population_Size-1
        for j-1:String_Length
                chromosome_change(i,j)=chromosome(selected(1,i),j);
        end
        fitness_function(1,i)=fitness_function(1,selected(1,i));
end

%select before crossover
for i=1:Population_Size-1
        x=round(rand(1)*32767);
        sum=round(sum_fit);
        rr=rem(x,sum)+1;
        n=1;
        partsum=0;
        while((partsum<rr)&(n<=Population_Size-1))
                partsum=partsum+fitness_function(1,n);
                bba=n;
                n=n+1;
        end
        selected(1,i)=bba;
end
%crossover
maxsw=max(fitness_function);
for i=1:Population_Size/2-1
        parent1=selected(1,i);
        parent2=selected(1,Population_Size-1-i);
        child1=i;
```

```
                child2 = Population _ Size - 1 - i;
                pc = 0.8;
                randnum = rand(1);
                site1 = round(rand(1) * String _ Length);
                for j = 1:String _ Length
                    if(j<site1)
                        chromosome(child1,j) = chromosome _ change(parent1,j);
                        chromosome(child2,j) = chromosome _ change(parent2,j);
                    else
                        chromosome(child1,j) = chromosome _ change(parent2,j);
                        chromosome(child2,j) = chromosome _ change(parent1,j);
                    end
                end
            end
            %mutation
            pm = 0.05;
            for i = 1:Population _ Size - 1
                for j = 1:String _ Length
                    randnum = rand(1);
                    if(randnum<pm)
                        chromosome(i,j) = chromosome(i,j);
                    end
                end
            end
        end
        generation = generation + 1;
    end
```

8.3　基于仿真的滤波器优化设计

滤波器优化设计的计算过程非常复杂、繁琐，但若采用 MATLAB 仿真技术，其过程可得到大大简化。由于 MATLAB 中功能强大的信号处理工具箱的出现和不断完善，使得滤波器的优化设计技术更为成熟。这种基于仿真技术的优化方法是一种新优化方法，在今后的实际应用中将会得到更大的关注和发展。

8.3.1　IIR 滤波器优化设计

设计 IIR 数字滤波器一般有两种方法：

① 变换法。首先设计一个合适的模拟滤波器，然后再变换成满足预定指标的数字滤波器。这种方法很方便，这是因为模拟滤波器已经具有很多简单而又现成的

设计公式,并且设计参数也已经表格化了,设计起来既方便又准确。

② 计算机辅助设计法。这是一种最优设计法。首先确定一种最优准则,例如设计出的实际频率响应幅度$|H(e^{j\omega})|$与所要求的理想频率响应幅度$|H_d(e^{j\omega})|$的均方误差最小准则,或它们的最大误差最小准则等;然后求在此最佳准则下滤波器系统函数的系数。这种设计一般得不到滤波器系数所要求的理想频率响应的闭合形式的函数表达式,而是需要进行大量迭代运算。

虽然变换法简单、方便,但也存在一些不足,如变换过程繁琐不易掌握,设计结果往往并不理想,查询表中提供的数据有限等。因此,现代先进的优化技术不断地用来设计更好的 IIR 滤波器。下面介绍基于 MATLAB 仿真技术和遗传算法的 IIR 数字滤波器的优化设计。

根据均方误差最小准则,在一组离散的频率点 $\omega_i(i=1,2,\cdots,M)$上所要求的频率响应 $H_d(e^{j\omega})$的值为$|H_d(e^{j\omega})|$,假定实际求出的频率响应为 $H(e^{j\omega})$,那么在这些给定的离散频率点处,所要求的频率响应的幅度与求出的实际频率响应的幅度的均方误差为:

$$E=\sum_{i=1}^{M}[|H(e^{j\omega_i})|-|H_d(e^{j\omega_i})|]^2 \tag{8-5}$$

优化设计的目的是调整各 $H(e^{j\omega})$,即调整 $H(e^{j\omega})$的系数,使 E 最小。这样即可将得到的 $H(e^{j\omega})$作为 $H_d(e^{j\omega})$的逼近值。

实际滤波器 $H(e^{j\omega})$常采用二阶节的级联形式表示,因为这种结构的频率响应对系数变化的灵敏度较低,便于调整频率响应。设:

$$H(z)=A\prod_{k=1}^{N}\frac{1+a_kz^{-1}+b_kz^{-2}}{1+c_kz^{-1}+d_kz^{-2}} \tag{8-6}$$

则滤波器的频率响应为:

$$H(e^{j\omega})=A\prod_{k=1}^{N}\frac{1+a_k^{-j\omega}+b_k^{-j\omega}}{1+c_k^{-j\omega}+d_k^{-j\omega}}=AP(e^{j\omega}) \tag{8-7}$$

其中:

$$P(e^{j\omega})=\prod_{k=1}^{N}\frac{1+a_k^{-j\omega}+b_k^{-j\omega}}{1+c_k^{-j\omega}+d_k^{-j\omega}} \tag{8-8}$$

滤波器在各频率采样点处的均方误差可表示为:

$$E=\sum_{i=1}^{M}[|H(e^{j\omega_i})|-|H_d(e^{j\omega_i})|]^2 \tag{8-9}$$

将(8-7)式代入(8-9)式可得:

$$E=\sum_{i=1}^{M}[|A|\cdot|P(e^{j\omega_i})|-|H_d(e^{j\omega_i})|]^2 \tag{8-10}$$

在(8-7)式中,共有$(4k+1)$个未知参量,将 A 以外的 $4k$ 个参量表示成矢量为:

$$\boldsymbol{\Phi}=(a_1,b_1,c_1,d_1,a_2,b_2,c_2,d_2,\cdots,a_k,b_k,c_k,d_k)^T \tag{8-11}$$

最佳增益 A 可用 E 对 A 求微分的方法求出,将 E 对 A 求微分并令其为 0,得:

$$\frac{\partial E}{\partial |A|}=\sum_{i=1}^{M}\{2[|A|\cdot|P(e^{j\omega_i})|-|H_d(e^{j\omega_i})|]|P(e^{j\omega_i})|\}=0 \tag{8-12}$$

于是,可以得到最佳增益 A:

$$|A|=\frac{\sum_{i=1}^{M}|P(e^{j\omega_i})|\cdot|H_d(e^{j\omega_i})|}{\sum_{i=1}^{M}|P(e^{j\omega_i})|^2} \tag{8-13}$$

由于只考虑滤波器的幅度响应误差,所以 A 的正负值对结果没有影响。为了保证所设计出的滤波器是稳定的,需要对优化参数的取值范围进行限定,同时也可使参数的取值范围得以缩小以提高精度。由(8-6)式可知,只要使每个二阶节的极点都在 Z 平面的单位圆内,即使 $1+c_kz^{-1}+d_kz^{-2}(k=1,2,\cdots,N)$的零点满足条件$|z_k|<1$,就可得参数 c_k、d_k 的取值范围为:

$$-2<c_k<2,\quad -1<d_k<1,\quad k=1,2,\cdots,N \tag{8-14}$$

若参数在此范围内取值,优化结果中仍出现了不稳定的极点,那么可用其倒数来代替该极点,从而在不改变幅频响应的情况下,保证得到稳定的 IIR 数字滤波器。如果将 a_k 和b_k 的取值范围也限定在Z 平面的单位圆内,则所设计出的滤波器是具有最小相位的滤波器。

例 8-4　设计一低通 IIR 数字滤波器,性能指标为:

$$|H_d(e^{j\omega})|=\begin{cases}1 & 0\le\omega\le0.4\pi\\ 0 & 0.5\pi\le\omega\le\pi\end{cases}$$

选取滤波器的阶数为 6,频率采样点取为 46,待优化参数共有 12 个,为了保证设计的滤波器是稳定的且具有最小相位,参数的取值范围选取如(8-14)式所示。采用遗传算法进行优化,在 MATLAB 中编程运行,即可得到优化结果。所得 IIR 数字滤波器的传递函数为:

$$H(z)=0.062261\,\frac{1+1.161752z^{-1}+0.993481z^{-2}}{1+0.464307z^{-1}-0.091974z^{-2}}\times\frac{1+0.204855z^{-1}+0.9899534z^{-2}}{1-0.426702z^{-1}+0.760436z^{-2}}\times\frac{1+1.880238z^{-1}+0.882931z^{-2}}{1-0.373876z^{-1}+0.2568162z^{-2}} \tag{8-15}$$

滤波器的频率响应曲线如图 8-1 所示。

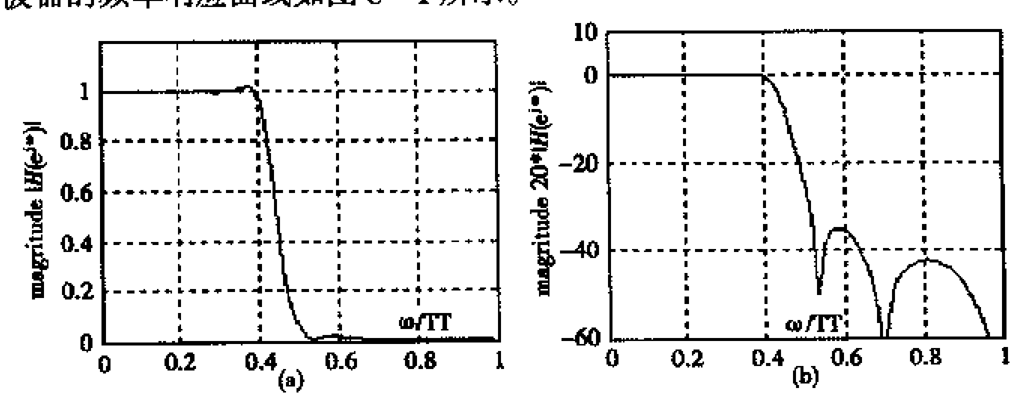


图 8-1　IIR 滤波器的频率响应

8.3.2 FIR 滤波器优化设计

FIR 数字滤波器设计的常用方法包括窗口法、频率抽样法、最优化设计法等。频率抽样法可以直接在频率域进行 FIR 滤波器设计，且效果很好，对于频率响应只有少数几个非零值抽样的窄带选频滤波器特别有效。但使用频率抽样法存在如何确定过渡带样本值的问题。传统的方法是查表取得，但查表取得却出现了新的问题即表中提供的数据不能保证是最优的，表中不可能提供关于不同的抽样点数、通带边缘频率、阻带边缘频率、过渡样本数的全部数据。若表中查不到的数据，只能通过近似估计确定过渡带样本值，这样不一定能保证全局最优。为了解决使用频率抽样法设计 FIR 滤波器存在的缺点，下面介绍基于 MATLAB 仿真技术和遗传算法的 IIR 数字滤波器优化设计。

设所设计的 FIR 数字滤波器的频率响应是 $H_d(e^{j\omega})$，它是频率 ω 的周期函数，对它进行抽样，使得每一个周期有 N 个抽样值，即：

$$H_d(k)=H_d(e^{j\omega})\big|_{\omega_k=\frac{2\pi}{N}k}=H_d(e^{j\frac{2\pi}{N}k}) \qquad (8-16)$$

$H_d(k)$可以表示为：

$$H_d(k)=H_g(k)e^{j\varphi_d(k)} \qquad (8-17)$$

式中 $H_g(k)$是滤波器的增益，$\Psi_d(k)$是其相位响应。

首先对 $H_d(k)$作离散 Fourier 逆变换，得到 N 点单位抽样响应序列：

$$h(n)=\frac{1}{N}\sum_{k=0}^{N-1}H_d(k)e^{j\frac{2\pi}{N}kn} \qquad n=0,1,\cdots,N-1 \qquad (8-18)$$

然后根据 Z 变换构成滤波器的转移函数：

$$H(z)=\sum_{n=0}^{N-1}h(n)z^{-n} \qquad (8-19)$$

滤波器对应的频率特性是：

$$H(e^{j\omega})=\sum_{n=0}^{N-1}h(n)e^{jn\omega} \qquad (8-20)$$

经过傅立叶变换可得到：

$$H(e^{j\omega})=e^{-j(N-1)\omega/2}\sum_{k=0}^{N-1}H_d(k)e^{j(N-1)k\pi/N}\frac{\sin[N(\omega-2\pi k/N)/2]}{N\sin[(\omega-2\pi k/N)/2]} \qquad (8-21)$$

例 8－5　给定低通滤波器的技术指标：通带边缘频率 $\omega p=0.2\pi$，阻带边缘频率 $\omega s=0.3\pi$，最大通带波动 Ap＝0.4dB，最小阻带衰减 As＝60dB。用频率采样技术设计一个 FIR 滤波器。

频率抽样点数为 N＝60，在过渡带取两个样本点 dot1，dot2∈(0,1)。采用遗传算法进行优化，在 MATLAB 中编程运行，得到优化结果：dot1＝0.5919034、dot2＝0.1074485、Ap＝0.22dB、As＝66.57dB，幅度响应曲线如图 8－2(a)所示。为了加以对比，给出查表法所得结果为：dot1＝0.59250、dot2＝0.10990、Ap＝0.3062dB、As＝63.4742dB，幅度响应曲线如图 8－2(b)所示。

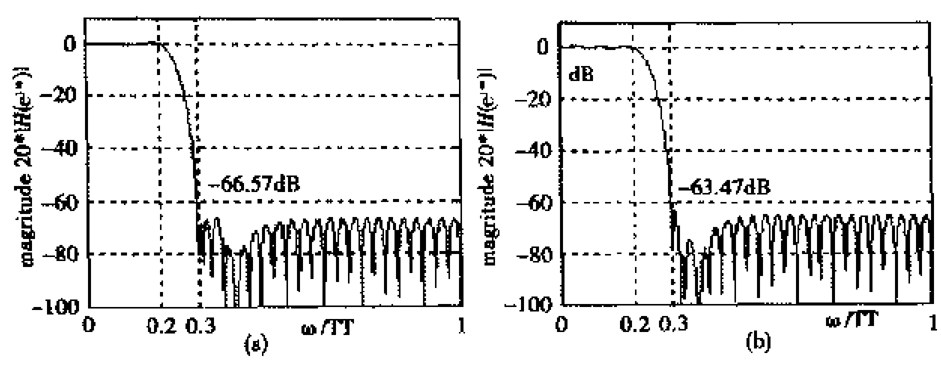


图 8－2　例 8－5 中滤波器的幅度响应曲线

本节所介绍的 IIR 数字滤波器和 FIR 数字滤波器优化设计方法不只是对低通滤波器有效，还完全可以用于设计其他类型的滤波器，如带通、高通、高阻滤波器，甚至可以用来优化设计任意的或多带的滤波器。

8.4　基于仿真的控制系统优化设计

MATLAB 语言所具有的强大的科学计算功能和方便、快捷、直观的图形仿真技术为控制系统的设计带来了巨大的便利。它可用于控制系统中十分复杂的计算，可用于单输入、单输出控制系统和多输入、多输出控制系统的动态仿真，可用于进行控制系统的优化设计。由于 MATLAB 在控制系统中的应用面很广，这里不能一一介绍。本节只重点介绍基于现代优化技术——遗传算法的控制系统优化设计，以体现 MATLAB 仿真技术的高级应用。

8.4.1　PID 控制器优化设计

控制系统的分析是在系统结构和参数已知的前提下，研究系统的特性与参数之间的关系，而控制系统的设计是根据对控制系统特性的要求，选择合适的控制方案与系统结构、计算参数及元器件，并通过仿真和实验研究，一直到建立能满足要求的实用系统为止。这里只讨论技术性部分，即用数学方法寻找一个能满足技术要求的控制系统，通常把这项工作叫做系统的综合。

控制系统大体上分为受控系统和控制器两大部分。受控系统是系统的基本部分，在设计过程中是不可变部分，但一般来说这部分的性能较差，难以满足对系统提出的要求，甚至是不稳定的，所以必须引入附加装置加以校正。控制器的重要组成部分是校正装置，因此，综合的中心是校正，综合的具体任务是选择校正的方式、确定系统结构的形式和校正装置的类型以及计算参数等。

控制系统的校正装置种类繁多，但就校正功能而言，它们是统一的，可分为三种基本类型：超前校正装置、迟后校正装置和迟后－超前校正装置。

◆　**超前校正装置**

超前校正装置的传递函数可描述为：

$$G(s)=\frac{1+\alpha T_d s}{1+T_d s} \tag{8-22}$$

其中,参数 $\alpha>1$,表示超前的"深度",是超前校正装置的分度系数,T_d 为超前校正装置的时间常数。设计超前校正装置的最终任务就是确定参数 α 和 T_d。通常,将以(8-22)式所示的超前校正装置为主体的控制器称为比例、微分控制器,即PD控制器。

应用超前校正装置对控制系统进行串联校正时,利用的是它所提供的超前角,以补偿控制系统因惯性而造成的相角迟后,从而有效地改善系统的特性。

◆ **迟后校正装置**

迟后校正装置的传递函数可描述为:

$$G(s)=\frac{1+\beta T_i s}{1+T_i s} \tag{8-23}$$

其中,参数 $\beta<1$,是迟后校正装置的分度系数,表示迟后的"深度",T_i 为迟后校正装置的时间常数。设计迟后校正装置的最终任务就是确定参数 β 和 T_i。通常将以(8-23)式所示的迟后校正装置为主体的控制器称为比例、积分控制器,即PI控制器。

应用迟后校正装置对控制系统进行串联校正时,利用的是校正装置的高频幅值衰减特性,它不仅可以提高系统的抗干扰能力,而且可以改善系统的特性。

◆ **迟后-超前校正装置**

迟后-超前校正装置的传递函数可描述为:

$$G(s)=\frac{1+\beta T_i s}{1+T_i s}\cdot\frac{1+\alpha T_d s}{1+T_d s} \tag{8-24}$$

其中,参数 $\alpha>1$,$\beta<1$,且 $T_i\gg T_d$。通常将以(8-24)式所示的迟后-超前校正装置为主体的控制器称为比例、积分、微分控制器,即PID控制器。设计迟后-超前校正装置的最终任务就是确定参数 α、β 和 T_d、T_i。

将迟后与超前装置组合在一起,取长补短,可构成性能较完善的校正装置。应用它进行控制系统的串联校正时,主要是利用超前校正部分产生的超前角和迟后部分所具有的高频幅值衰减特性,使校正后的系统在稳态和暂态两方面都具有较大的改善。

综上所述,引入PD控制器进行超前校正,可以有效地改善系统的暂态性能,但对稳态性能的改善则很有限;而引入PI控制器进行迟后校正,则可在维持原有满意的暂态性能的同时,有效地提高系统的稳态性能。因此,将两者组合起来,组成PID控制器用于对系统进行迟后-超前校正,是同时满足对暂态与稳态性能均要求较高的一种常用校正方式。本节就以设计PID控制器为例讨论MATLAB在控制系统优化设计中的应用。

设计控制器的任务是确定其中所包含的各个参数的取值,传统的方法是采用根轨迹校正法、状态空间的极点配置法等。虽然采用这些设计方法设计出的控制器在一定程度上基本能满足系统的性能指标要求,但却难以得到更好的控制器;但若采用现代优化技术对其进行优化设计,则可得到性能更优越的控制器。例8-6采用遗传算法对控制系统的控制器进行优化设计,这种方法是以MATLAB仿真技术为基础的。

例8-6 控制系统结构如图8-3所示,$Gc(s)$为控制器,$Gg(s)$为控制对象。原算例要求:静态速度误差系数 $Kv\geqslant100s-1$,最大超调量 $\sigma\%\leqslant20\%$,调节时间 ts≤3 S。控制对象 $Gg(s)$为:

$$G_g(s)=\frac{1000}{s(s+10)(s+100)}$$

根据系统要求，采用迟后－超前校正装置，控制器 $Gc(s)$ 为：

$$G_c(s)=\frac{K(T_2s+1)(\alpha T_1s+1)}{(\beta T_2s+1)(T_1s+1)}$$

其中，K、T_1、T_2、α、β 均为可调参数，$1<\alpha\leqslant\beta$，$T_2\geqslant\alpha T_1$。

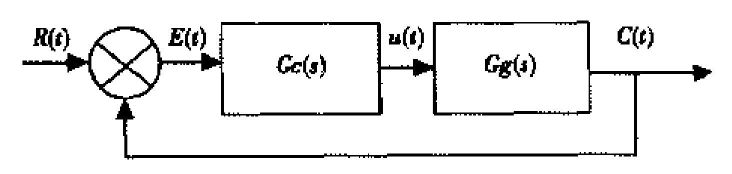


图8－3　例8－6中控制系统的结构图

从迟后－超前校正装置的传递函数可知，需要优化的参数有五个，比例增益 K，时间常数 T_1、T_2，比例系数 α、β，它们的组合构成了控制器设计的一个可行解 $x=(K,T_1,T_2,\alpha,\beta)$。五个参数各有其独立的变化范围，$K\in[K_d,K_u]$、$T_1\in[T_{d1},T_{u1}]$、$T_2\in[T_{d2},T_{u2}]$、$\alpha\in[\alpha_d,\alpha_u]$、$\beta\in[\beta_d,\beta_u]$，并需满足约束条件 $1<\alpha\leqslant\beta$，$T_2\geqslant\alpha T_1$。而所设计出的控制系统的性能指标是最大超调量，响应时间以及稳态误差等。因为各参数都是在其相应的连续区域中变动，组合方式是无穷的，要想找出多个目标衡量这5个参数的最优组合，即问题的最优解 $x_{op}=(K_{op},T_{op},T_{2op},\alpha_{op},\beta_{op})$，其复杂度是可想而知的。传统的优化算法难以获得参数的满意组合，而采用遗传算法是一种较好的解决途径。

在MATLAB中编写出遗传算法程序，并经调试后，即可得到满意的结果。以下为三组较好的结果：

- 参数 $\{K,T_1,T_2,\alpha,\beta\}=\{162.3021, 0.0147, 0.2516, 8.5572, 12.9658\}$，相应的控制系统阶跃响应曲线如图8－4中曲线1所示。
- 参数 $\{K,T_1,T_2,\alpha,\beta\}=\{149.8387, 0.0152, 10.0128, 8.5308, 8.7801\}$，相应的控制系统阶跃响应曲线如图8－4中曲线2所示。
- 参数 $\{K,T_1,T_2,\alpha,\beta\}=\{105.3763, 0.0226, 0.6026, 5.3372, 6.0606\}$，相应的控制系统阶跃响应曲线如图8－4中曲线3所示。

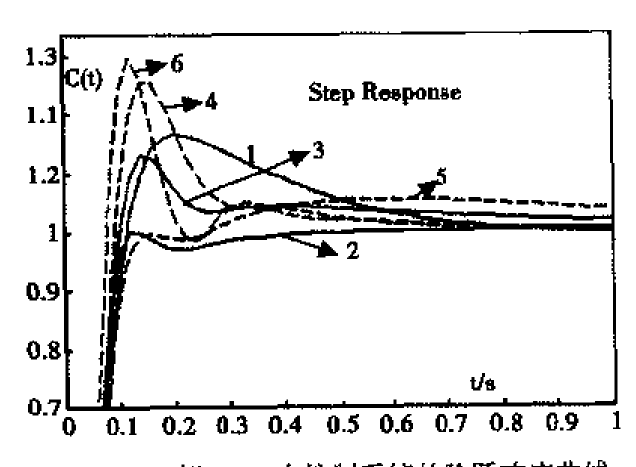


图8－4　例8－6中控制系统的阶跃响应曲线

作为对比，列出参考文献[15]中给出的3个控制器，分别为：

- 参数 $\{K,T_1,T_2,\alpha,\beta\}=\{100.0, 0.022, 0.25, 4.60, 5\}$，相应的控制系统阶跃

响应曲线如图 8－4 中曲线 4 所示。

- 参数$\{K, T_1, T_2, \alpha, \beta\} = \{100.0, 0.02, 0.5, 10.0, 10.0\}$，相应的控制系统阶跃响应曲线如图 8－4 中曲线 5 所示。
- 参数$\{K, T_1, T_2, \alpha, \beta\} = \{100.0, 0.025, 0.31, 4.60, 4.0\}$，相应的控制系统阶跃响应曲线如图 8－4 中曲线 6 所示。

从图 8－4 可以看出，采用优化方法设计的控制器具有更令人满意的综合性能指标。图 8－4 中曲线 1、曲线 2 和曲线 3 与曲线 4、曲线 5 和 6 相比，系统响应速度更快，上升时间和调节时间都比参考文献[15]中的三组设计短，系统超调量更小，不仅完全满足所确定的性能指标要求，而且结果表明，其设计是相当令人满意的。

8.4.2 二次型控制器优化设计

线性二次型(LQ)最优控制已成为现代控制理论及应用中最富有成果的一部分。线性二次型性能指标易于分析、计算、处理，可得到要求的代数结果，通过线性二次型最优化方法得到的闭环系统具有鲁棒性好，无穷大增益裕量和 60°相角裕量的优点，因而在控制理论的各个领域得到了广泛地重视和应用。

由于线性二次型调节器(LQR)控制的闭环系统的动态响应与加权系数矩阵 Q 和 R 之间存在着非常复杂的对应关系，这就给加权矩阵的选择带来许多困难，目前普遍采用的仿真试凑法无疑限制了 LQR 设计方法在工程上的推广应用。采用试凑法获得的最优控制是“人工”意义下的最优，并不是真正意义上的最优。实际上在这种情况下无法获得最优解，因而采用基于仿真的优化方法求取满意解是解决这类问题的最佳途径。

下面以三级倒立摆为例来讨论二次型控制器的优化设计问题。三级倒立摆系统很复杂，用经典的 PID、根轨迹校正和极点配置法难以实现其控制，采用 LQ 法又无经验数据可查，只能采用仿真试凑法来确定加权系数矩阵，但这是非常复杂的过程。采用基于仿真的优化方法对 LQR 进行优化设计，可以很好地解决加权系数矩阵确定困难的问题。

例 8－7 三级倒立摆系统是一个单输入多输出的八阶控制系统，其结构原理图、各参数含义及取值和非线性数学模型请参见参考文献[16]，将其非线性数学模型在 $\gamma = \theta_1 = \theta_2 = \theta_3 = 0, \dot{\gamma} = \dot{\theta}_1 = \dot{\theta}_2 = \dot{\theta}_3 = 0$ 不稳定平衡点附近进行线性化，进行线性化后所得到的数学模型如下：

$$\begin{cases} \dot{X} = AX + Bu \\ Y = CX \end{cases}$$

其中，$X = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)^T = [\gamma, \theta_1, \theta_2 - \theta_1, \theta_3 - \theta_2, \dot{\gamma}, \dot{\theta}_1, \dot{\theta}_2 - \dot{\theta}_1, \dot{\theta}_3 - \dot{\theta}_2]^T$，$Y = (y_1, y_2, y_3, y_4)^T$，$D$ 为零矩阵，A、B、C 的取值分别为：

$$A = \begin{bmatrix} A_{11} & A_2 \\ A_{21} & A_{22} \end{bmatrix}, B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, C = [C_1 \quad C_2],$$

其中，A_{11} 为四阶零矩阵，A_{12} 为四阶单位矩阵，$A_{21} = \begin{bmatrix} 0 & -1.14 & 0.53 & -0.04 \\ 0 & 34.34 & -48.8 & 3.123 \\ 0 & -41.36 & 129.2 & -21.5 \\ 0 & 8.66 & -98.3 & 69.08 \end{bmatrix}$，

$$A_{22}=\begin{bmatrix}-16.67 & 0.013 & -0.01 & 0.001\\ 41.078 & -0.22 & 0.173 & -0.11\\ -49.49 & 0.46 & -0.43 & 0.394\\ 10.44 & -0.28 & 0.39 & -0.73\end{bmatrix}$$，B_1 为 4×1 的零矢量，$B_2=\begin{bmatrix}8.64\\ -21.29\\ 25.66\\ -5.41\end{bmatrix}$，

C_1 为四阶单位矩阵，C_2 为四阶零矩阵。

三级倒立摆控制系统的结构框图如图 8－5 所示。

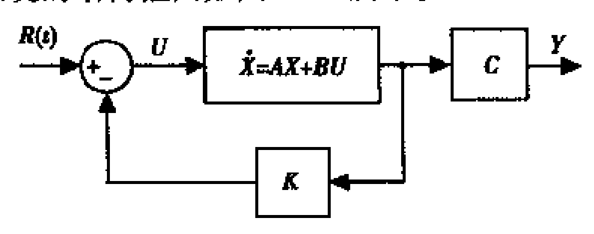


图 8－5　例 8－7 中系统的结构图

由参考文献[16]可知，三级倒立摆控制系统是可控可观的，但却是一个不稳定的系统。系统待优化的参数共有 9 个，令 $Q=\mathrm{diag}(q_1,q_2,q_3,q_4,q_5,q_6,q_7,q_8)$，$R=q_9$，于是优化问题的解向量 $q=[q_1,q_2,q_3,q_4,q_5,q_6,q_7,q_8,q_9]$。

本例采用基于仿真技术的优化方法对该控制系统的 LQ 控制器进行优化设计，将算法用 MATLAB 编程实现，经过调试后即可得到满意的结果。以下为两组较好的结果：

- 参数 $[q_1,q_2,q_3,q_4,q_5,q_6,q_7,q_8,q_9]=[9.3549\quad 28.0352\quad 421.0166\quad 111.3392\quad 0.0293\quad 0.1075\quad 0.0782\quad 3.9345\quad 0.0157]$，经计算得到的状态反馈阵为：$K=[24.4\quad 274.7\quad 81.5\quad 1101.8\quad 39.2\quad 136.3\quad 126.3\quad 153.9]$，其阶跃响应曲线如图 8－6(a)所示。
- 参数 $[q_1,q_2,q_3,q_4,q_5,q_6,q_7,q_8,q_9]=[9.4234\quad 20.5572\quad 284.9462\quad 160.9971\quad 0.1271\quad 0\quad 0.1417\quad 3.8123\quad 0.0284]$，经计算得到的状态反馈矩阵为：$K=[18.2002\quad 199.3881\quad 60.1183\quad 769.6124\quad 29.3883\quad 97.5579\quad 89.0252\quad 107.5777]$，其阶跃响应曲线如图 8－6(b)所示。

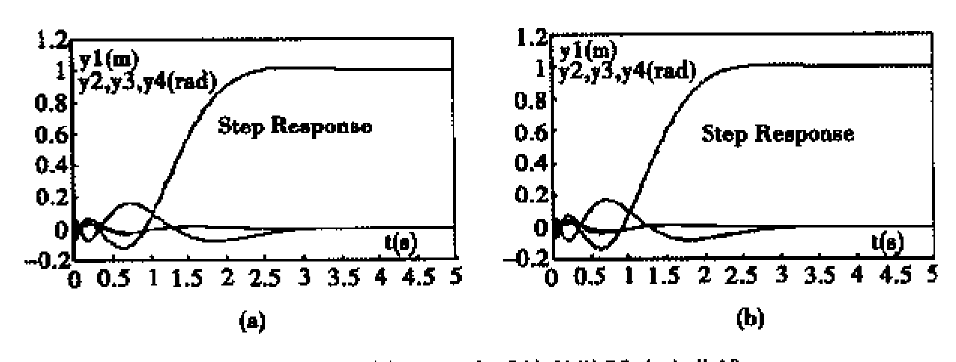


图 8－6　例 8－7 中系统的阶跃响应曲线

三级倒立摆控制系统是一个很复杂的强不稳定系统，不过，从仿真结果可以看出，采用优化方法设计的三级倒立摆控制器能使系统具有满意的综合性能指标，图 8－6 中三级倒立摆系统的两响应曲线显示，系统各性能指标均满足设计要求，系统的过渡过程时间比较短，超调量也较小。

8.5 本章小结

本章主要介绍了 MATLAB 仿真技术在优化领域中的应用。首先给出了优化问题的一般数学描述，然后对线性规划和非线性规划问题的 MATLAB 求解方法进行了介绍，为了便于读者进一步学习本章第 3 节和第 4 节，本章在第 2 节中对遗传算法进行了简单介绍，最后重点讨论了在 MATLAB 中利用遗传算法进行 FIR 滤波器、IIR 滤波器、PID 控制器和 LQR 控制器的优化设计问题。

习　题

1. 求解下面的四元线性规划问题：

$$\max\{-x_1+2x_2-x_3+3x_4\}$$

$$s.t.\quad \begin{cases} x_1+x_2+3x_3+x_4=6 \\ -2x_1+x_3+x_4\leq 3 \\ -x_1+6x_3-3x_4\leq 4 \\ x_1,x_2,x_3,x_4\geq 0 \end{cases}$$

2. 求解下面的四元二次规划问题：

$$\min\{(x_1-1)^2+(x_2-2)^2+(x_3-3)^2+(x_4-4)^2\}$$

$$s.t.\quad \begin{cases} x_1+x_2+x_3+x_4\leq 5 \\ 3x_1+3x_2+2x_3+x_4\leq 10 \\ x_1,x_2,x_3,x_4\geq 0 \end{cases}$$

3. 求解下面的有约束条件最优化问题：

$$\min\{1000-x_1^2-2x_2^2-x_3^2-x_1x_2-x_1x_3\}$$

$$s.t.\quad \begin{cases} x_1^2+x_2^2+x_3^2=25 \\ 8x_1+14x_2+7x_3=56 \\ x_1,x_2,x_3\geq 0 \end{cases}$$

4. 编制一个可执行的遗传算法程序，并思考如何改进算法的性能？

5. 设计一个 8 阶低通 IIR 数字滤波器，其性能指标为：

$$|H_d(e^{j\omega})|=\begin{cases} 1 & 0\leq\omega\leq 0.2\pi \\ 0 & 0.3\pi\leq\omega\leq\pi \end{cases}$$

6. 用频率采样法设计一个 FIR 低通滤波器，其性能指标为：通带边缘频率 $\omega p=0.5\pi$，阻带边缘频率 $\omega s=0.6\pi$，最大通带波动 Ap = 0.4dB，最小阻带衰减

As = 60dB。

7. 对图 8-7 中的随动系统进行补偿，用优化方法设计一个 PID 控制器，使之具有如下的性能指标：

(1) 单位斜坡响应的稳态误差小于 0.09。

(2) 相角裕量为 40°。

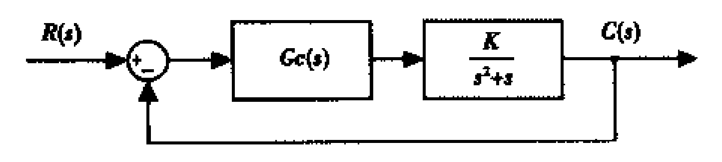


图 8-7　习题 7 中待补偿系统结构图

附录 A　MATLAB 常用函数

本附录搜集了 MATLAB 绝大多数常用的函数及语句命令，以方便用户查询。同时，由于本书主要介绍的是 MATLAB 在控制系统中的应用，因此将控制系统工具箱中的命令也在此列出。其他各工具箱的命令和函数请参阅附录 B。

表 A1　通用命令

命令和函数类别	命令和函数名称	命令和函数说明
管理命令和函数	HELP	在线帮助文件
	DOC	装入超文本说明
	WHAT	M、MAT、MEX 文件的目录列表
	TYPE	列出 M 文件
	LOOKFOR	通过 HELP 条目搜索关键字
	WHICH	定位函数文件
	DEMO	运行演示程序
	PATH	控制 MATLAB 的搜索路径
管理变量和工作空间	WHO	列出当前变量
	WHOS	列出当前变量(长表)
	LOAD	从磁盘文件中恢复变量
	SAVE	保存工作空间变量
	CLEAR	从内存中清除变量和函数
	PACK	整理工作空间内存
	SIZE	矩阵的尺寸
	LENGTH	向量的长度
	DISP	显示矩阵或文本
与文件和操作系统有关的命令	CD	改变当前工作目录
	DIR	目录列表
	DELETE	删除文件
	GETENV	获取环境变量值
	!	执行 DOS 操作系统命令
	UNIX	执行 UNIX 操作系统命令并返回结果
	DIARY	保存 MATLAB 任务
控制命令窗口	CEDIT	设置编辑命令快捷键
	CLC	清空命令窗口
	HOME	光标置左上角
	FORMAT	设置输出格式
	ECHO	底稿文件内使用的回显命令
	MORE	在命令窗口中控制分页输出

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
启动和退出 MATLAB	QUIT	退出 MATLAB
	STARTUP	启动 MATLAB 时自动执行的 M 文件
	MATLABRE	主启动 M 文件
一般信息	INFO	MATLAB 系统信息及 MATHWORKS 公司信息
	SUBSCRIBE	成为 MATLAB 的定购用户
	HOSTID	MATLAB 主服务程序的识别代号
	WHATSNEW	在说明书中未包含的新信息
	VER	版本信息

表 A2　操作符和特殊字符

命令和函数类别	命令和函数名称	命令和函数说明
操作符和特殊字符	+	加
	-	减
	*	矩阵乘法
	.*	数组乘法
	^	矩阵幂
	.^	数组幂
	\	左除或反斜杠
	/	右除或斜杠
	./	数组除
	KRON	KRONECKER 张量积
	:	冒号
	()	圆括号
	[]	方括号
	.	小数点
	..	父目录
	...	继续
	,	逗号
	;	分号
	%	注释
	!	感叹号
	'	转置或引用
	=	赋值
	= =	相等
	< >	关系操作符
	&	逻辑与
	|	逻辑或
	~	逻辑非
	XOR	逻辑异或
逻辑函数	EXIST	检查变量或函数是否存在
	ANY	向量的任一元为真，则其值为真
	ALL	向量的所有元为真，则其值为真
	FIND	找出非零元素的索引号

表 A3　基本数学函数

命令和函数类别	命令和函数名称	命令和函数说明
三角函数	SIN	正弦
	SINH	双曲正弦
	ASIN	反正弦
	ASINH	反双曲正弦
	COS	余弦
	COSH	双曲余弦
	ACOS	反余弦
	ACOSH	反双曲余弦
	TAN	正切
	TANH	双曲正切
	ATAN	反正切
	ATAN2	四象限正切
	ATANH	反双曲正切
	SEC	正割
	SECH	双曲正割
	ASECH	反双曲正割
	CSC	余割
	CSCH	双曲余割
	ACSC	反余割
	ACSCH	反双曲余割
	COT	余切
	COTH	双曲余切
	ACOT	反余切
	ACOTH	反双曲余切
指数函数	EXP	指数
	LOG	自然对数
	LOG10	常用对数
	SQRT	平方根
复数函数	ABS	绝对值
	ARGLE	相角
	CONJ	复共扼
	IMAGE	复数虚部
	REAL	复数实部
数值函数	FIX	朝零方向取整
	FLOOR	朝负无穷大方向取整
	CEIL	朝正无穷大方向取整
	ROUND	朝最近的整数取整
	REM	除后余数
	SIGN	符号函数

表 A4　基本矩阵和矩阵函数

命令和函数类别	命令和函数名称	命令和函数说明
基本矩阵	ZEROS ONES EYE RAND RANDN LOGSPACE MESHGRID :	零矩阵 全“1”矩阵 单位矩阵 均匀分布的随机数矩阵 正态分布的随机数矩阵 对数间隔的向量 三维图形的 X 和 Y 数组 规则间隔的向量
特殊变量和常数	ANS EPS REALMAX REALMIN PI I ,J INF NAN FLOPS NARGIN NARGOUT COMPUTER ISIEEE WHY VERSION	当前的答案 相对浮点精度 最大浮点数 最小浮点数 圆周率值 3.141 592 653 589 7…… 虚数单位 无穷大 非数值 浮点运算次数 函数输入变量数 函数输出变量数 计算机类型 当计算机采用 IEEE 算术标准时,其值为真 简明的答案 MATLAB 版本号
时间和日期	CLOCK DATE ETIME TIC TOC EPUTIME	墙上挂钟 日历 计时函数 秒表开始执行 计时函数 CPU 时间(以秒为单位)
矩阵操作	DIAG FLIPLR FLIPUD RESHAPE ROT90 TRIL TRIU :	建立或提取对角矩阵 矩阵作左右翻转 矩阵作上下翻转 改变矩阵大小 矩阵旋转 90° 提取矩阵的下三角部分 提取矩阵的上三角部分 矩阵的索引号,重新排列矩阵

表 A5 特殊矩阵

命令和函数类别	命令和函数名称	命令和函数说明
特殊矩阵	COMPAN	友矩阵
	HADAMARD	HADAMARD 矩阵
	HANKEL	HANKEL 矩阵
	HILB	HILBERT 矩阵
	INVHILB	逆 HILBERT 矩阵
	KRON	KRONECKER 张量积
	MAGIC	魔方矩阵
	TOEPLITZ	TOEPLITZ 矩阵
	VANDER	VANDERMONDE 矩阵

表 A6 矩阵函数——数值线性代数

命令和函数类别	命令和函数名称	命令和函数说明
矩阵分析	COND	计算矩阵条件数
	NORM	计算矩阵或向量范数
	RCOND LINPACK	逆条件值估计
	RANK	计算矩阵秩
	DET	计算矩阵行列式值
	TRACE	计算矩阵的迹
	NULL	零矩阵
	ORTH	正交化
线性方程	\ 和/	线性方程求解
	CHOL	CHOLESKY 分解
	LU	高斯消元法求系数矩阵
	INV	矩阵求逆
	QR	正交三角矩阵分解(简称 QR 分解)
	PINV	矩阵伪逆
特征值和奇异值	EIG	求特征值和特征向量
	POLY	求特征多项式
	HESS	HESSBERG 形式
	QZ	广义特征值
	CDF2RDF	变复对角矩阵伪实分块对角形式
	SCHUR	SEHUR 分解
	BALANCE	矩阵均衡处理以提高特征值精度
	SVDE	奇异值分解
矩阵函数	EXPM	矩阵指数
	EXPM1	实现 EXPM 的 M 文件
	EXPM2	通过泰勒技术求矩阵指数
	EXPM3	通过特征值和特征向量求矩阵指数
	LOGM	矩阵对数
	SQRTM	矩阵开平方根
	FUNM	一般矩阵的计算

表 A7　泛函——非线性数值方法

命令和函数类别	命令和函数名称	命令和函数说明
泛函——非线性数值方法	ODE23	低阶法求解常微分方程
	ODE23P	低阶法求解常微分方程并绘出结果图形
	ODE45	高阶法求解常微分方程
	QUAD	低阶法计算数值积分
	QUAD8	高阶法计算数值积分
	FMIN	单变量函数的极小化
	FMINS	多变量函数的极小化
	FZERO	找出单变量函数的零点
	FPLOT	函数绘图

表 A8　多项式函数

命令和函数类别	命令和函数名称	命令和函数说明
多项式函数	ROOTS	求多项式根
	POLY	构造具有指定根的多项式
	POLYVALM	带矩阵变量的多项式计算
	RESIDUE	部分分式展开(留数计算)
	POLYFIT	数据的多项式拟和
	POLYDER	微分多项式
	CONV	多项式乘法
	DECONV	多项式除法

表 A9　通用图形函数

命令和函数类别	命令和函数名称	命令和函数说明
建立和控制图形窗口	FIGURE	建立图形(图形窗口)
	GEF	获取当前图形的句柄
	ELF	清除当前图形
	CLOSE	关闭图形
建立和控制坐标系	SUBPLOT	在标定位置上建立坐标系
	AXES	在任意位置上建立坐标系
	GCA	获取当前坐标系的句柄
	CLA	清除当前坐标系
	AXIS	控制坐标系的刻度和形式
	CAXIS	控制伪色彩坐标刻度
	HOLD	保持当前图形
句柄图形对象	FIGURE	建立图形窗口
	AXES	建立坐标系
	LINE	建立曲线
	TEXT	建立文本串
	PATCH	建立图形填充块
	SURFACE	建立曲面
	IMAGE	建立图像

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
句柄图形对象	UICONTROL UIMEN	建立用户界面控制 建立用户界面菜单
句柄图形操作	SET· GET RESET DELETE NEWPLOT GCO DRAWNOW FINDOBJ	设置对象 获取对象特征 重置对象特征 删除对象 预测 NEXTPLOT 性质的 M 文件 获取当前对象的句柄 填充未完成绘图事件 寻找指定特征值的对象
打印和存储	PRINT PRINTOPT ORIENT CAPTURE	打印图形或保存图形 配置本地打印机默认值 设置纸张取向 屏幕抓取当前图形

表 A10　二维图形函数

命令和函数类别	命令和函数名称	命令和函数说明
基本 X－Y 图形	PLOT LOGLOG SEMILOGX SEMILOGY FILL	线性图形 对数坐标图形 半对数坐标图形(X 轴为对数坐标) 半对数坐标图形(Y 轴为对数坐标) 绘制二维多边形填充图形
特殊 X－Y 图形	POLAR BAR STEM STAIRS ERRORBAR HIST ROSE COMPASS FEATHER FPLOT COMET	极坐标图 条形图 离散序列图或杆图 阶梯图 误差条图 直方图 角度直方图 区域图 箭头图 绘图函数 星点图
图形注释	TITLE XLABEL YLABEL TEXT GTEXT GRID	图形标题 X 轴标记 Y 轴标记 文本注释 用鼠标放置文本 网格线

表 A11　语言结构

命令和函数类别	命令和函数名称	命令和函数说明
MATLAB 编程语言	FUNCTION	增加新的函数
	EVAL	执行由 MATLAB 表达式构成的字串
	FEVAL	执行由字串指定的函数
	GLOBAL	定义全局变量
程序控制流	IF	条件执行语句
	ELSE	与 IF 命令配合使用
	ELSEIF	与 IF 命令配合使用
	END	FOR, WHILE 和 IF 语句的结束
	FOR	重复执行指定次数(循环)
	WHILE	重复执行不定次数(循环)
	BREAK	终止循环的执行
	RETURN	返回引用的函数
	ERROR	显示信息并终止函数执行
交互输入	INPUT	提示用户输入
	KEYBOARD	像底稿文件一样使用键盘输入
	MENU	产生由用户输入选择的菜单
	PAUSE	等待用户响应
	UIMENU	建立用户界面菜单
	UICONTROL	建立用户界面控制

表 A12　字符串函数

命令和函数类别	命令和函数名称	命令和函数说明
一般函数	STRINGS	MATLAB 中有关字符串函数的说明
	ABS	变字符串为数值
	SETSTR	变数值为字符串
	ISSTR	当变量为字符串时其值为真
	BLANKS	空字符串
	DEBLANK	删除尾部的空串
	STR2MAT	从各个字符串中形成文本矩阵
	EVAL	执行由 MATLAB 表达式组成的串
字符串比较	STRCMP	比较字符串
	FINDSTR	在一字符串中查找另一个子串
	UPPER	变字符串为大写
	LOWER	变字符串为小写
	ISLETTER	当变量为字母时，其值为真
	ISSPACE	当变量为空白字符时，其值为真
字符串与数值之间变换	NUM2STR	变数值为字符串
	INT2STR	变整数为字符串
	STR2NUM	变字符串为数值
	SPRINTF	变数值为格式控制下的字符串
	SSCANF	变字符串为格式控制下的数值
十进制与十六进制之间变换	HEX2NUM	变十六进制数为 IEEE 标准下的浮点数
	HEX2DEC	变十六进制数为十进制数
	DEC2HEX	变十进制数为十六进制数

附录 B　MATLAB 常用工具箱函数

本附录收集了 MATLAB 中常用的工具箱函数及语句命令，以方便读者查询。事实上，用户可以自己编写一些 MATLAB 函数，然后按照某种特定的要求生成自己的工具箱。表 B-1 为工具箱目录索引，其余各表为工具箱函数的具体内容。

表 B1　工具箱目录索引

工具箱名称	工具箱内容	表 索 引
SIGNAL	信号处理	表 B-2
COMMUNICATION	通信系统	表 B-3
CONTROL	控制系统	表 B-4
OPTIM	最优化	表 B-5
FUZZY	模糊系统	表 B-6
IDENT	系统辨识	表 B-7
IMAGE	图像处理	表 B-8
LOCAL	局部函数库	表 B-9
NCD	非线性控制设计	表 B-10
NNET	神经网络	表 B-11
ROBUST	鲁棒控制	表 B-12

表 B2　信号处理工具箱

命令和函数类别	命令和函数名称	命令和函数说明
波形产生	SAWTOOTH	产生锯齿波或三角波
	SQUARE	产生方波
	SINC	产生 SINC 或函数
	DIRIC	产生 DIRICHLET 或周期 SINC 函数
滤波器分析和实现	ABS	取绝对值(幅值)
	ANGLE	取相角
	CONV	求卷积
	FLTFILT	重叠相加法 FFT 滤波器实现
	FILTER	直接滤波器实现
	FILTFILT	零相位数字滤波
	FILTIC	FILTER 函数初始条件选择
	FREQS	模拟滤波器频率响应
	FRSQSPACE	频率响应中的频率间隔
	FREQZ	数字滤波器频率响应
	GRPDELAY	平均滤波延迟(群延迟)
	IMPZ	数字滤波器的冲激响应
	ZPLANE	离散系统零极点图

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
线性系统变换	CONVMTX	卷积矩阵
	POLY2RE	从多项式系数中计算反射系数
	RESIDUEZ	Z 变换部分分式展开或留数计算
	RE2POLY	从反射系数中计算多项式系数
	SOS2SS	变系统二阶分隔形式为状态空间形式
	SOS2TF	变系统二阶分隔形式为传递函数形式
	SOS2ZP	变系统二阶分隔形式为零极点增益形式
	SS2SOS	变系统状态空间形式为二阶分割形式
	SS2TF	变系统状态空间形式为传递函数形式
	SS2ZP	变系统状态空间形式为零极点增益形式
	TF2SS	变系统传递函数形式为状态空间形式
	TF2ZP	变系统传递函数形式为零极点增益形式
	ZP2SOS	变系统零极点增益形式为二阶分割形式
	ZP2SS	变系统零极点增益形式为状态空间形式
	ZP2TF	变系统零极点增益形式为传递函数形式
IIR 滤波器设计	BESSELF	BESSEL(贝塞尔)模拟滤波器设计
	BUTTER	BUTTERWORTH(比特沃思)滤波器设计
	CHEBY1	CHEBYSHEV(切比雪夫)Ⅰ型滤波器设计
	CHEBY2	CHEBYSHEV(切比雪夫)Ⅱ型滤波器设计
	ELLIP	椭圆滤波器设计
	YULEWALK	递归数字滤波器设计
FIR 滤波器设计	FIR1	基于窗函数的 FIR 滤波器设计——标准响应
	FIR2	基于窗函数的 FIR 滤波器设计——任意响应
	FIRLS	最小二乘 FIR 滤波器设计
	INTFILT	内插 FIR 滤波器设计
	REMEZ	PARKS－MCCELLAN 最优 FIR 滤波器设计
	REMEZORD	PARKS－MCCELLAN 最优 FIR 滤波器阶数选择
变换	CZT	线性调频 Z 变换
	DET	离散余弦变换(DCT)
	IDET	逆离散余弦变换
	DFTMTX	离散傅立叶变换矩阵
	FFT	一维快速傅立叶变换
	IFFT	一维逆快速傅立叶变换
	FFTSHIFT	重新排列 FFT 的输出
	HILBERT	希尔伯特变换
统计信号处理	COV	协方差矩阵
	XCOV	互协方差函数估计
	CORRCOEF	相关系数矩阵
	XCORR	互相关函数估计

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
统计信号处理	COHERE	相关函数平方幅值估计
	CSD	互谱密度(CSD)估计
	PSD	信号功率谱密度(PSD)估计
	TFE	从输入输出中估计传递函数
窗函数	BOXCAR	矩形窗
	TRIANG	三角窗
	BARTLETT	BARTLETT(巴特利特)窗
	HAMMING	HAMMING(哈明)窗
	HANNING	HANNING(汉宁)窗
	BLACKMAN	BLACKMAN(布莱克曼)窗
	CHEBWIN	CHEBYSHEV 窗
	KAISER	KAISER 窗
参数化建模	INVFREQS	模拟滤波器拟合频率响应
	INVFREQZ	离散滤波器拟合频率响应
	PRONY	利用 PRONY 方法的离散滤波器拟合时间响应
	STMCB	利用 STEIGLITZ－MEBRIDE 迭代方法求线性模型
	LEVINSON	LEVINSON－DURBIN 递归算法
	LPC	线性预测算法
特殊操作	RCEPS	实倒谱和最小相位重构
	CCEPS	倒谱分析和最小相位重构
	ATE	降低序列的取样速率
	INTERP	提高取样速率(内插)
	RESAMPLE	改变取样速率
	MEDFILT1	一维中取滤波
	DECONV	反卷积和多项式除法
	MODULATE	通信仿真中的调制
	DEMOD	通信仿真中的解调
	VCO	电压控制振荡器
	SPECGRAM	频谱分析
模拟原型滤波器设计	BUTTAP	BUTTERWORTH 模拟低通滤波器原型
	CHEB1AP	CHEBYSHEV Ⅰ 型模拟低通滤波器原型
	CHEB2AP	CHEBYSHEV Ⅱ 型模拟低通滤波器原型
	ELLIPAP	椭圆模拟低通滤波器原型
	BESSELAP	BESSEL 模拟低通滤波器原型
频率变换	LP2BP	低通到带通模拟滤波器变换
	LP2HP	低通到高通模拟滤波器变换
	LP2BS	低通到带阻模拟滤波器变换
	LP2LP	低通到低通模拟滤波器变换
滤波器离散化	BILINEAR	双线性变换
	IMPRINVAR	冲激响应不变法实现模拟到数字滤波器变换

(续表)

命令和函数类别	命令和函数名称	命令和函数说明
其他	CONV2	二维卷积
	CPLXPAIR	将复数归成复共轭对
	DETREND	删除线性趋势
	FFT2	二维快速傅立叶变换
	IFFT2	二维逆快速傅立叶变换
	FITER2	二维数字滤波器
	PILYSTAB	稳定多项式
	XCORR2	二维互相关

表 B3 通信系统工具箱

命令和函数类别	命令和函数名称	命令和函数说明
信号源	RANDERR	产生误比特数图样
	RANDINT	产生均匀分布的随机整数矩阵
	RANDSRC	产生指定符号的随机矩阵
	WGN	产生高斯白噪声
	AWGN	产生加性高斯白噪声
信号分析函数	BITERR	误比特数及误比特率计算
	EYEDIAGRAM	产生眼图
	SCATTERPLOT	产生散布图
	SYMERR	误比特数和误符号率计算
信源编/译码函数	COMPAND	M 律或 A 律压扩计算
	DPCMDECO	差分脉码调制译码
	LLOYDS	使用 LLOYD 算法优化量化参数
	QUANTIZ	产生量化序号和量化值
纠错控制编码/解码函数	BCHPOLY	产生二进制 BCH 码的参数或生成多项式
	CONVENC	卷积码编码计算
	CYCLGEN	产生循环码的生成矩阵和校验矩阵
	DECODE	分组码译码器
	ENCODE	分组码编码器
	CYCLPOLY	产生循环码的生成多项式
	GENZPAR	生成矩阵和校验矩阵之间的转换
	GFWEIGHT	计算线性分组码的最小码距
	HAMMGEN	产生海明码的生成矩阵和校验矩阵
	RSDECOF	对已编码的 ASCII 文件用 R－S 码译码
	RSENCOF	对 ASCII 文件进行 R－S 码编码
	RSPOLY	产生 R－S 码的生成多项式
	SYNDTABLE	产生综合译码表
	VITDEC	用 VITERBI 算法进行卷积译码

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
纠错编/译码底层函数	BCHDECO	BCH 译码
	BCHENCO	BCH 编码
	RSDECO	R－S 码译码
	RSDECODE	用指数形式进行 R－S 译码
	RSENCO	R－S 码编码
	RSENCODE	用指数形式进行 R－S 编码
调制/解调函数	ADEMOD	模拟通带解调
	ADEMODCE	模拟基带解调
	AMOD	模拟通带调制
	AMODCE	模拟基带调制
	APKCONST	给出 ASK－PSK 调制的环形星座图
	DDEMOD	数字通带解调
	DDEMODCE	数字基带解调
	DEMODMAP	模拟信号到数字信号的映射
	DMOD	数字通带调制
	DMODCE	数字基带调制
	MODMAP	数字信号到模拟信号的映射
	QASKDECO	矩形星座 QASK 的逆映射
	QASKENCO	绘制 QASK 矩形星座图
特殊滤波器函数	HANKZSYS	将 HANKEL 矩阵转换成线性系统模型
	HILBIIR	HILBERT、IIR 滤波器设计
	RCOSFLT	对输入信号进行升余弦滤波
	RCOSINE	升余弦滤波器设计
特殊滤波器底层函数	RCOSFIR	升余弦 FIR 滤波器设计
	RCOSIIR	升余弦 IIR 滤波器设计
实用工具箱函数	BIZDE	二进制向量转换成十进制向量
	DEZBI	十进制向量转换为二进制向量
	ERF	误差函数
	ERFC	互补误差函数
	ISTRELLIS	格型结构的有效性检验
	MARCUMQ	广义 Q 函数
	OCTZDEC	八进制转换为十进制
	POLYZTRELLIC	将卷积码多项式转换为格型描述
	RECZMAT	将向量转换为矩阵
GALOIS 域计算函数	GFADD	多项式加法计算
	GFCONV	多项式乘法计算
	GFCOSETS	割园陪集计算
	GFDECONV	逆卷积计算
	GFDIV	除法计算

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
GALOIS 域计算函数	GFFILTER	多项式数据过滤计算
	GFLINEQ	方程 AX=B 求解
	GFMINPOL	寻找最小多项式
	GFMUL	将向量转换为矩阵
	GFPLUS	加法计算
	GFPRETTY	多项式表示
	GFPRIMCK	多项式的可约性检测
	GFPRIMCLF	输出指定维数的原始多项式
	GFPRIMFG	寻找原始多项式
	GFRANK	计算矩阵的秩
	GFREPCON	多项式表示方式的转换
	GFROOTS	求多项式的根
	GFSUB	减法计算
	GFTRUNC	多项式最短化处理
	GFTUPLE	多项式的约简和转化

表 B4　控制系统工具箱

命令和函数类别	命令和函数名称	命令和函数说明
建模	CLOOP	系统的闭环
	CONNECT	方框图建模
	CONV	两个多项式的卷积
	FEEDBACK	反馈系统连接
	ORD2	产生二阶系统的 A、B、C、D
	PARALLEL	并行系统连接
	SERIES	串行系统连接
模型变换	POLY	变更值表示为多项式表示
	RESIDUE	部分分式展开
	SS2TF	变状态空间表示为传递函数表示
	SS2ZP	变状态空间表示为零极点表示
	TF2SS	变传递函数表示为状态空间表示
模型简化	BALREAL	平衡实现
	DBALREAL	离散平衡实现
	DMODRED	离散模型降阶
	MINERAL	最小实现和零极点对消
	MODRED	模型降阶
	TF2ZP	变传递函数表示为零极点表示
	ZP2TF	变零极点表示为传递函数表示
	ZP2SS	变零极点表示为状态空间表示

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
模型实现	CANON CTRBF OBSVF SS2SS	正则形式 可控阶梯形 可观阶梯形 采用相似变换
模型特性	CTRB DAMP DEGAIN EIG OBSV ROOTS	可控性矩阵 阻尼系数和固有频率 连续稳态(直流)增益 特征值和特征向量 可观性矩阵 多项式之根
时域响应	DLSIM DSTEP IMPULSE INITIAL LSIM STEP STEPFUN	任意输入下的离散时间仿真 离散时间阶跃响应 冲激响应 连续时间零输入响应 任意输入下的连续时间仿真 阶跃响应 阶跃函数
频域响应	FBODE FREQS FREQZ MARGIN NICHOLS NYQUIST	连续系统的快速 BODE 图 拉普拉斯变换频率响应 Z 变换频率响应 增益和相位裕度 NICHOLS 图 NYQUIST 图
根轨迹	PZMAP RLOCUS	零极点图 画根轨迹
增益选择	ACKER LQR LQRY PLACE	单输入单输出极点配置 线性二次调节器设计 输出加权的调节器设计 极点配置
方程求解	ARE DLYAP LYAP LYAP2	代数 RICCATI 方程 离散 LYAPUNOV 方程求解 连续 LYAPUNOV 方程求解 利用对角化求解 LYAPUNOV 方程

表 B5 最优化工具箱

命令和函数类别	命令和函数名称	命令和函数说明
非线性最小化函数	ATTGOAL	达到多目标
	CONSTR	约束极小化
	FMIN	无约束极小化(标量情况)
	FMINU	利用梯度搜索的无约束极小化
	FMINS	利用单纯形搜索的无约束极小化
	FSOLVE	非线性方程求解
	LEASTSP	非线性最小二乘
	MINIMAX	极小极大求解
	SEMINF	半定极小化
矩阵问题极小化	LP	线性规划
	QP	二次规划
	NNLS	非负最小二乘
控制缺省值和选项	FOPTIONS	参数设置
演示	DATDEMO	数据拟合成曲线
	OPTDEMO	演示菜单
	TUTDEMO	启动教程
	BANADEMO	香蕉型函数的极小化
	GOALDEMO	目标达到
	DFILDEMO	有限精度滤波器设计
三次内插程序	CUBIC	内插 4 点,以找出极大值
	CUBICIL	内插 1 点和梯度,以估计极小值
	CUBICI2	内插 2 点和梯度,以找出步长和极小值
	CUBICI3	内插 3 点和梯度,以找出步长和极小值
	CUBICI4	内插 4 点和梯度,以找出步长和极小值
二次内插程序	QUAD2	内插 3 点,以找到最大值
	QUADINTER	内插 3 点,以估计最小值
演示实用程序	EIGFUN	返回分类特征值的函数
	ELIMONE	消去一变量
	FILTFUN	频率响应和根
	FILTFUN2	频率响应范数和根
	FITFUN	返回拟合数据中的误差范数
	FITFUN2	返回拟合数据中的误差矢量
半定实用程序	SEMIFUN	半定问题转换成约束问题
	FINDMAX	在数据向量中内插极大值
	FINDMAX2	在数据矩阵中内插极大值
	V2SORT	分类两向量,然后删去丢失的元素
目标达到的实用程序	GOALFUN	目标达到问题转换成约束条件问题
	GOALGRA	变换目标达到问题中的梯度

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
测试程序	TOPTIM TOPTIMF TOPTIMG	最优化测试组 最优化测试组的测试函数 最优化测试组的测试函数梯度
其他	GRADER LSINT OPTINT SEARCHQ	用于检查梯度的不一致性 初始化最小二乘程序的函数 初始化无约束极小程序的函数 线性搜索程序

表 B6　模糊系统工具箱函数

命令和函数类别	命令和函数名称	命令和函数说明
GUI 编辑器	FUZZY MFEDIT RULEEDIT RELEVIEW SURFVIEW	基本 FIS(模糊推理系统)编辑器 隶属度函数编辑器 规则编辑器及(句法)分析程序 规则观察器及模糊推理框图 输出曲面观测器
隶属度函数	DSIGMF GAUSS2MF GAUSSMF GBELLMF PIMF PSIGMF SMF SIGMF TRAPMF TRIMF ZMF	两个"S"形隶属度函数的差 双边高斯曲线隶属度函数 高斯曲线隶属度函数 广义钟形隶属度函数 π 形隶属度函数 两个"S"形隶属度函数的积 "S"形隶属度函数 "SIGMOID(S)"形隶属度函数 梯形隶属度函数 三角形隶属度函数 "Z"形隶属度函数
命令行 FIS 函数	ADDMF ADDRULE ADDVAR DEFUZZ EVALFIS EVALMF GENSURF GETFIS MF2MF NEWFIS PARSRULE PLOTFIS PLOTMF READFIS RMMF	将隶属度函数加到 FIS 中 将规则加到 FIS 中 将变量加到 FIS 中 去模糊隶属度函数 完成模糊推理计算 隶属度函数计算 产生 FIS 输出曲面 获得模糊系统的特性 在函数之间变换参数 产生新的 FIS 分析模糊规则 显示 FIS 输入/输出图 显示出一个变量的所有隶属度函数 从磁盘中装入 FIS 从 FIS 中删除隶属度函数

(续表)

命令和函数类别	命令和函数名称	命令和函数说明
命令行FIS函数	RMVAR	从FIS中删除变量
	SETFIS	设置模糊系统特征
	SHOWFIS	显示带注释的FIS
	SHOWRULE	显示FIS规则
	WRITEFIS	在磁盘中保存FIS
高级技术	ANFIS	SUGENO-TYPE FIS的训练程序
	FEM	利用C平均聚集方法找出簇
	GENFIS1	利用一般方法产生FIS矩阵
	GENFIS2	利用减法聚集方法产生FIS矩阵
	SUBCLUST	利用减法聚集方法估计簇中心

表B7 系统辨识工具箱函数

命令和函数类别	命令和函数名称	命令和函数说明
仿真和预测	IDSOM	仿真一给定的系统
	PE	计算预测误差
	POLY2TH	从给定的多项式中构造矩阵
	PREDICT	M步超前预测
数据处理	DTREND	从数据集中删除方位
	IDFILT	通过BUTTERWORTH滤波器对数据进行滤波
非参数化估计	COVF	估计数据矩阵的协方差矩阵
	CRA	相关分析
	ETFE	估计经验传递函数并计算周期图
	SPA	频谱分析
参数估计	AR	利用各种方法计算AR信号模型
	ARMAX	ARMAX模型预测误差估计
	ARX	ARX模型的最小二乘估计
	BJ	BOX-JENKINS模型的预测误差估计
	CANSTART	具有初值参数估计的多变量模型
	IVAR	时间序列的AR部分的仪器IV估计
	IVX	单输出ARX模型的仪器可变估计
	IV4	ARX模型近似最优的IV估计
	OE	输出误差模型的预测误差估计
	PEM	一般线性模型的预测误差估计
建立模型结构	ARX2TH	ARX模型的格式
	CANFORM	正则形模型结构
	MF2TH	将用户定义的模型结构封装入模型结构中
	MODSTRUC	在MS2TH函数中使用的模型结构
	MS2TH	将标准状态空间参数封装入格式中
	POLY2TH	从给定多项式中产生矩阵

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
处理模型结构	FIXPAR	在状态空间 ARX 模型结构中找出要修正的参数
	SETT	在结构中设置取样间隔
	THINIT	参数的(随机)初始值
	UNFIXPAR	在状态空间和 ARX 模型结构中释放参数
模型变换	TH2ARX	变格式模型为 ARX 模型
	TH2FF	求模型的频率响应及标准偏差
	TH2PAR	变格式为参数和协方差阵
	TH2POLY	求给定模型相应的多项式
	TH2SS	变格式为状态空间表示
	TH2TF	变格式为传递函数表示
	TH2ZP	求零极点　静态增益和标准偏差
	THE2THD	变连续时间模型为离散时间模型
	THD2THE	变离散时间模型为连续时间模型
模型表示	BODEPLOT	传递函数的 BODE 图或频谱
	FFPLOT	频域函数
	IDPLOT	输入/输出数据
	NYQPLOT	传递函数的 NYQUIST 图
	PRESENT	频谱上的参数模型
	ZPPLOT	零点和极点
信息提取	GETMFTH	获取定义模型结构的 M 文件名
	GETNEAP	获取数据点数和参数个数
	GETFF	选取频率函数
	GETT	为某模型获取取样间隔
	GETZP	从 TH2ZP 函数产生的零极点格式中提取零点和极点
模型合法化	COMPARE	将仿真和预测的输出与测量输出比较
	IDSIM	仿真一给定的系统
	PE	预测误差
	PREDICT	M 步超前预测
	RESID	计算和测试与某模型相关的留数
估计模型不确定性	IDSIMSD	在仿真模型响应中说明不确定性
	TH2FF	模型频率函数和标准偏差
	TH2ZP	零点、极点、静态增益及其标准偏差
模型结构选择	ARXSTRUC	ARX 模型类的损失函数
	IVSTRUC	单输出类的输出误差拟合
	SELSTRUC	根据各种准则选择模型结构
	STRUC	ARXSTRUC 和 IVSTRUC 的典型结构矩阵

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
递归参数估计	RARX	对 ARX 模型递归计算估值
	RARMAX	对 ARMAX 模型递归计算估值
	RBJ	对 BOX－JENKINS 模型递归计算估值
	ROE	对输出误差模型递归计算估值
	RPEM	对一般模型递归计算估值
	RPLR	对一般模型递归计算 PLR 估值
	SEGMENT	分断数据并跟踪快变系统

表 B8　图像处理工具箱函数

命令和函数类别	命令和函数名称	命令和函数说明
图像输入/输出	BMPREAD	从磁盘中读 BMP(WINDOWS 下的位图)文件
	BMPWRITE	将一 BMP 文件写入磁盘
	GIFREAD	从磁盘中读 GIF 文件
	GIFWRITE	将 GIF 文件写入磁盘
	HDFPEEK	从 HDF 文件中列出目标标记/参考对
	HDFREAD	从 HDF 文件中读取数据
	HDFWRITE	写数据到 HDF 文件中
	PEXREAD	从磁盘中读取 PEX 文件
	PEXWRITE	将 PEX 文件写入磁盘
	TIFFREAD	从磁盘中读取 TIFF 文件
	TIFFWRITE	将 TIFF 文件写入磁盘
	XWDREAD	从磁盘中读取 XWD 文件
	XWDWRITE	将 XWD 文件写入磁盘
实用程序	GETIMAGE	从坐标系中读取图像数据
	SBW	当图像为黑白图像时，其值为真
	ISGRAY	当图像为灰度图像时，其值为真
	ISIND	当图像为加标图像时，其值为真
颜色操作	BRIGHTEN	加亮或增暗一颜色板
	CMUNIQUE	寻找惟一的颜色板及相应的图像
	CMPERNUTE	置换颜色板位置
	CMGAMMA	校正颜色板
	CMGAMDEF	缺省的校正表
	DITHER	FLOYD－STEINBERG 图像颤抖算法
	HSV2RGB	变 HSV 值为 RGB 颜色空间
	IMADJUST	调整并增强图像强度
	IMAPPROX	利用更少颜色的图像逼近加标图像
	NTSC2RGB	变 NTSC 值为 RGB 颜色空间
	RGB2GRAY	变 RGB 值为灰度值
	RGB2HSV	变 RGB 值为 HSV 颜色空间
	RGB2NTSC	变 RGB 值为 NTSC 颜色空间
	RGBPLOT	控制 RGB 颜色板分量的图形

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
几何操作	IMCROP	修剪图像
	IMRESIZE	改变图像大小
	IMROTATE	旋转图像
	TRUESIZE	改变图像大小使之具有实际尺寸
	IMZOOM	放大或缩小图像和二维图形
图像增强/分析	BRIGHTEN	增强或削弱颜色板
	GRADSLICE	密度(强度)限幅
	HISTEQ	直方图均衡化
	IMADJUST	调整和展宽图像强度
	IMAPPROX	利用较少颜色的图像逼近图像
	IMHIST	图像直方图
	IMPIXEL	一像素点的颜色
	IMPROFILE	轮廓强度
	IMTERP2	二维数据内插
图像统计	MEAN2	矩阵的均值
	CORR2	二维相关系数
	STD2	二维标准差
形态操作	BWAREA	二进制图像中的目标区域
	DILATE	加浓二进制图像
	ERODE	冲淡二进制图像
	EDGE	边界提取
	BWEULER	欧拉数
	BWMORPH	通过频率取样的二维图像
	BWPERIM	二进制图像中目标周围
FIR(有限冲激响应)滤波器设计	FSAMP2	通过频率取样的二维 FIR 滤波器设计
	FSPECIAL	特殊的二维 FIR 滤波器设计
	FTRANS2	通过频率变换的二维 FIR 滤波器设计
	FWIND1	使用一维窗函数的 FIR 滤波器设计
	FWIND2	使用二维窗函数的 FIR 滤波器设计
	IMNOISE	图像噪声
频率响应	FREQSPACE	二维频率响应的频率空间
	FREQZ2	二维频率响应
滤波	COLFILT	局部非线性滤波
	CONV2	二维卷积
	FILTER2	二维滤波
	MEDFILT2	二维中值滤波
	MFILTER2	屏蔽滤波
	NLFILTER	局部非线性滤波
	WIENER2	自适应二维去噪滤波

(续表)

命令和函数类别	命令和函数名称	命令和函数说明
分块处理	BEATBLK	分块处理的最佳块大小
	BLKPROC	按块处理一图像
	COL2IM	重新排列以形成图像
	COLFILT	局部非线性滤波
	IM2COL	重新排列成列
个别区域	MFILTER2	屏蔽滤波
	ROIPOLY	定义感兴趣的多边区域
	ROICOLOR	用颜色定义感兴趣的区域
变换	DCT2	二维离散余弦变换
	FFT2	二维快速傅立叶变换
	FFTSHIFT	零频率移到频率中心
	IDCT2	二维逆离散余弦变换
	IFFT2	二维逆快速傅立叶变换
	RADON	RADON 变换
转换	DITHER	FLOYD-STEINBERG 图像颤抖
	GRAY2IND	变灰度图像为加附标图像
	HSV2RGB	变 HSV 值为 RGB 值
	IM2BW	变图像为黑白图像
	IMSLICE	在图像中获取/置入图像块
	IND2GRAY	变附标图像为灰度图像
	IND2RGB	变附标图像为 RGB 图像
	MAT2GRAY	变矩阵为灰度图像
	NTSC2RGB	变 NTSC 值为 RGB 值
	RGB2GRAY	变 RGB 图像或值为灰度图像或值
	RGB2HSV	变 RGB 值为 HSV 值
	RGB2IND	变 RGB 图像为附标图像
	RGB2NTSC	变 RGB 值为 NTSC 值
图像显示	COLORBAR	显示颜色条
	COLORMAP	设置或获取颜色查找表
	GRAY	线性灰度颜色板
	HSV,HOT,JET	颜色板
	IMAGE	显示附标图像
	IMAGESC	数据定标并按图像显示
	IMCONTOUR	图像等高线
	IMMOVIE	制作图像动画
	IMSHOW	显示所有类型的图像数据
	MONTAGE	按矩形剪辑方式显示图像
	SUBIMAGE	显示多个图像
	WARP	将图像卷成曲面

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
演示	IMDEMO DCTDEMO FIRDEMO NLFDEMO	一般图像处理演示 二维离散余弦变换图像压缩演示 二维 FIR 滤波器演示 二维非线性滤波演示
专用函数	CUMSUM3D DCT DCTMTX2 DITHERE ELEM3D GETLINE GETPTS GETRECT GIF HDFREADC JDFPEEKC HDFWE IDCT IM2GRAY IMHISTC NDX3D RGB2IM RLE TIFF VMQUANT WAITBAR	三维矩阵封装成二维矩阵时的累加和 一维离散余弦变换 一元二维离散余弦变换矩阵 图像颤抖的 MEX 文件 将三维矩阵封装成二维矩阵的元素位置 利用橡皮线跟踪鼠标移动 利用可视点跟踪鼠标移动 利用橡皮矩阵跟踪鼠标移动 压缩 GIF 数据 读 HDF 文件的 MEX 文件 搜索 HDF 文件的 MEX 文件 写 HDF 文件的 MEX 文件 一维逆离散余弦变换 变图像为灰度图像 显示图像直方图 三维矩阵封装成二维矩阵的索引 变 RGB 图像为附标或强度图像 压缩数码数据 压缩 TIFF 编码数据 与彩色量化 MEX 文件接口的 M 文件 显示等待条
MAT 文件	BWMORPH. MAT FOREST. MAT MRI. MAT TREES. MAT	BWMORPH. M 文件的查找表 CARMANAH OLD GROWTH FOREST 的扫描相片 人体心脑的磁性共振图像 树的扫描图像

表 B9　局部函数库

命令和函数类别	命令和函数名称	命令和函数说明
常见函数	MATLABRC PRINTTOPT	MATLAB 的主启动 M 文件 设置打印选项

表 B10 非线性控制设计工具箱函数

命令和函数类别	命令和函数名称	命令和函数说明
对话框的管理	CONEDDLG	管理 NCD 工具箱固定编辑器的对话框
	PARAMDLG	管理 NCD 优化参数的对话框
	RANGEDLG	管理 NCD 坐标系范围的对话框
	REFDLG	管理 NCD 参考信号的对话框
	STEPDLG	管理 NCD 阶跃响应的对话框
	UNCERDLG	管理 NCD 不确定变量的对话框
主要界面	CONTRNCD	建立 NCD 固定图形的用户界面控制
	MENUNCD	建立 NCD 固定图形的用户界面菜单
	NCDBLOCK	包含 NCD 框图的 SIMULINK 系统
	OPTBLOCK	打开一个 NCD 图形的底稿文件
	OPTFIG	建立一个 NCD 固定图形
主要优化技术	COSTFUN	NCD 优化的代价函数
	NLINOPT	执行优化算法
演示实例	NCDDEMO	包含所有 NCD 演示实例的 SIMULINK 系统
	NCDDEMO1	PID 控制器
	NCDDEMO2	带前馈控制器的 LQR
	NCDDEMO3	多输入多输出的 PI 控制器
	NCDDEMO4	倒摆演示教程
	NCDTUT1	控制设计实例
	NCDTUT2	系统辨识实例
用户界面工具	DIALOG	创建对话框图像
	ERRORDLG	建立出错对话框
	FIGFLAG	图像显示在当前屏幕上时其值为真
	HELPDLG	显示帮助对话框
	LAYOUT	定义对话框布局参数的底稿文件
	QUEATDLG	建立提问对话框
	UIGUIDE	有关用户界面约定/标准/建议的说明
	WARNDLG	建立警告对话框
演示和教程工具	NCD1INIT	为 NCDDEMO1 的优化进行设置
	NCD2INIT	为 NCDDEMO2 的优化进行设置
	NCD3INIT	为 NCDDEMO3 的优化进行设置
	NCD4INIT	为 NCDDEMO4 的优化进行设置
	PENDDATE	为 NCDTUT2(即倒摆)进行设置
界面实用工具	CUROBJ	提供有关当前点的信息
	DEVIDECB	将固定界分为两部分
	DELINE	从 NCD 图中删除所有的图
	DONEP	收回 CLOSE 按钮和菜单
	ERRORNCD	管理 NCD 产生的常见错误
	FILLAXES	建立约束边界并进行数据检测
	FORCEIT	在已经存在的界限内插入一子集

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
界面实用工具	KEYNCD	NCD 按键函数
	LOADNCD	装入并显示 NCD 数据
	MAKESURF	建立并限界曲面
	SNAPNCD	以 22.5 间隔排出约束条
	REFRESHO	使约束矩阵与图像一致
	SAVELOAD	当前文件是从 SELECTFILE 中选择时，其值为真
	TEXTED	收回 PORT 可编辑文本
	UNDONCD	放弃上次 NCD 图形用户界面的操作
	UPDATDLG	更新 NCD 对话框
最优化实用工具	CONVERTM	变约束矩阵为最优化格式
	MINIIPARS	NCD 最小化分析
	MONTEVAR	初始化 MONFE CARLO 仿真
	NCDGLOB	定义 NCD 全局变量
	STR2MAT2	变一行字符串为多行字符串
帮助文本文件（以.HLP 为扩展名）	HOTKEY	热键帮助
	MAINNCD	一般 NCD 帮助
	PARAMDLG	最优化参数对话框帮助
	READNCD	与 README. M 文件内容相同
	STEPDLG	阶跃响应对话框的帮助
	UNCERDLG	不确定性变量对话框的帮助

表 B11　神经网络工具箱函数

命令和函数类别	命令和函数名称	命令和函数说明
误差分析函数	ERRSURF	计算误差曲面
	PLOTEP	在误差曲面上绘制权和基位置图
	PLOTES	绘制误差曲面图
δ 函数	DELTALIN	对 PURELIN 神经元的函数
	DELTALOG	对 LOGSIG 神经元的函数
	DELTATAN	对 TANSIG 神经元的函数
设计	SOLVEHOP	设计 HOPFIELD 网络
	SOLVELIN	设计线性网络
	SOLVERB	设计径向基网络
	SOLVERBE	设计精确的径向基网络
初始化	INITC	竞争层初始化
	INITEM	ELMAN 递归网络初始化
	INITFF	至多三层的前向网络初始化
	INITLIN	线性层初始化
	INITLVQ	LVQ 网络初始化
	INITP	感知层初始化
	INITSM	自组织映射初始化

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
初始化	MIDPOINT	产生中点值
	NWLOG	对LOGSIG神经元产生NGUYEN－WIDROW随机数
	NWTAN	对TANSIG神经元产生NGUYEN－WIDROW随机数
	RANDNC	产生归一化列随机数
	RANDNR	产生归一化行随机数
	RANDS	产生对称随机数
学习规则	LEARNBP	反向演播学习规则
	LEARNBPM	带预测的反向演播学习规则
	LEARNH	HEBB学习规则
	LEARNHD	退化的HEBB学习规则
	LEARNIS	内星学习规则
	LEARNK	KOHONEN学习规则
	LEARNLM	LEVENBERG－MARQUARDT学习规则
	LEARNLVP	学习矢量量化规则
	LEARNOS	外星学习规则
	LEARNP	感知层学习规则
	LEARNPN	归一化的感知层学习规则
	LEARNWH	WIDROW－HOFF学习规则
矩阵	COMBVEC	创建所有的矢量集
	DELAYSIG	从信号矩阵中建立退化的信号矩阵
	DIST	计算矢量距离
	IND2VEC	变下标矢量为稀疏矩阵表示
	NORMC	归一化矩阵列
	NORMR	归一化矩阵行
	PNOMC	伪归一化矩阵列
	QUANT	离散化成某数值的整数倍
	SUMSQR	平方和
	VECT2IND	变稀疏矩阵表示为下标矢量
邻域	NBDIST	使用矢量距离的邻域阵
	NBGRID	使用栅格距离的邻域阵
	NBMAN	使用MANHATTAN距离的邻域阵
绘图	BARER	每个输出矢量的误差条形图表
	HINTONW	绘制权值图
	HINTONWB	绘制权值和偏差图
	PLOTERR	绘制网络误差与时间关系图
	PLOTES	绘制误差曲面
	PLOTFA	绘出目标模式及网络函数的逼近
	PLOTPV	绘出限幅神经元的感知器分类
	PLOTSM	绘制自组织映射图
	PLOTTR	绘出网络误差记录及自适应学习速率
	PLOTVEC	用不同颜色绘制矢量

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
仿真	SIMUC	竞争层仿真
	SIMUELM	ELMAN 递归网络仿真
	SIMUFF	前向网络仿真
	SIMUHOP	HOPFIELD 网络仿真
	SIMULIN	线性层仿真
	SIMUP	感知层仿真
	SIMURB	径向基网络仿真
	SIMUSM	自组织映射仿真
训练	TRAINBP	利用反向演播训练前向网络
	PX	利用快速反向演播训练网络
	TRAINC	训练竞争层网络
	TRAINELM	训练 ELMAN 递归网络
	TRAINLVP	训练 LVQ 网络
	TRAINP	利用感知规则训练感知层
	TRAINPN	利用归一化感知规则训练感知层
	TRAINSM	利用 KOHONEN 规则训练自组织映射
	TRAINWH	利用 WIDROW－HOFF 规则训练线性层
传递函数	COMPET	竞争层传递函数
	HARDLIM	硬限幅传递函数
	HARDLIMS	对称硬限幅传递函数
	LOGSIG	对数 S 型传递函数
	PURELIN	线性传递函数
	RADBAS	径向基传递函数
	SATLINS	对称饱和线性传递函数
	TANSIG	正切 S 型传递函数

表 B12　鲁棒控制工具箱函数

命令和函数类别	命令和函数名称	命令和函数说明
可选系统数据结构	BRANCH	从树中提取一分支
	GRAFT	在树中增加一分支
	ISSYSTEM	辨识一系统变量
	ISTREE	辨识一树型变量
	MKSYS	为系统建立树变量
	TREE	建立树变量
	VRSYS	返回标准系统变量名
建模	AUGSS	系统增广（状态空间模型）
	AUGTF	系统增广（传递函数模型）
	INTERC	一般多变量内连系统

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
模型转换	BILIN	多变量双线性变换
	DES2SS	利用奇异值分解变系统为状态空间系统
	LFTF	线性分式变换
	SECTF	扇形变换
	STABPROJ	稳定和逆稳定映射
	SLOWFAST	慢/快分解
	TFM2SS	变传递函数模型为状态空间模型
实用工具	ARESOLV	广义连续时间 RICCATI 方程求解
	DARESOLV	广义离散时间 RICCATI 方程求解
	RICCOND	连续时间 RICCATI 方程的条件数
	DRICCOND	离散时间 RICCATI 方程的条件数
	BLKRSCH	得到实方阵的有序块实 Schur 形式
	CSCHUR	通过复旋转得到有序复 SCHUR 形式
多变量 BODE 图	CGLOCI	连续特性增益轨迹
	DCGLOCI	离散特性增益轨迹
	DSIGMA	离散奇异值 BODE 图
	MUOPT	具有实/复数混合不确定性系统的 SSV(结构化奇异值)上界
	OSBORNE	通过 OSBORNE 法求得的 SSV 上界
	PERRON	计算 PERRON 特征值
	PSV	PERRON 特征结构的 SSV
	SIGMA	连续奇异值 BODE 图
	SSV	结构化奇异值 BODE 图
因子分解技术	IOFC	内外因子分解(列类型)
	IOFR	内外因子分解(行类型)
	SFL	左边频谱分解
	SFR	右边频谱分解
模型简化方法	BALMR	截断均衡模型简化
	BSTSCHML	相对误差 SCHUR 模型简化
	IMP2SS	从脉冲响应到状态空间实现
	OBALREAL	有序均衡实现
	OHKLMR	最优 HANKEL 极小化逼近
	RECHUR	SCHUR 模型简化
鲁棒控制综合方法	H2LQG	H2 连续时间综合
	DH2LQG	H2 离散时间综合
	HINF	H_∞ 连续时间综合
	DHINF	H_∞ 离散时间综合
	HINFOPT	综合的迭代

（续表）

命令和函数类别	命令和函数名称	命令和函数说明
鲁棒控制综合方法	NORMH2	计算 2 范数
	NORMHINF	计算无穷范数
	LQG	LQG 最优控制综合
	LTRU	LQG 闭环 U 传递补偿
	LTRY	LQG 闭环 Y 传递补偿
	YOULA	YOULA 参数化
演示实例	ACCDEMO	弹簧质量标准问题
	DINTDEMO	双积分器系统的 H_∞ 设计
	HINFDEMO	飞机或大型空间结构的 H 或 H_∞ 设计实例
	LTRDEMO	LQR/LTR 设计实例：飞机
	MUDEMO	综合实例
	MRDEMO	鲁棒模型简化实例
	RCTDEMO	鲁棒控制工具箱演示—主菜单

参考文献

[1] 康凤举．现代仿真技术与应用．北京：国防工业出版社,2001
[2] 刘藻珍,魏华梁．系统仿真．北京：北京理工大学出版社,1998
[3] 范影乐,杨胜天,李轶．MATLAB 仿真应用详解．北京：人民邮电出版社,2001
[4] 薛定宇,陈阳泉．基于 MATLAB/SIMULINK 系统仿真技术与应用．北京：清华大学出版社,2002
[5] 张汉全,肖建,汪晓宁．自动控制理论新编教程．成都：西南交通大学出版社,2000
[6] 张家祥,罗雪山,方凌江．MATLAB 中虚拟现实的应用．电子计算机．2002,155(4):54－57
[7] 段哲民,范世贵．信号与系统．西安：西北工业大学出版社,2001
[8] 程佩青．数字信号处理教程(第二版)．北京：清华大学出版社,2001
[9] McClellan J H,Burrus C S,Oppenheim A V, etc. Computer－Based Exercises for Signal Processing Using MATLAB 5, Prentice－Hall. Inc,1998
[10] Lapsley P J,Shoham B A,etc. DSP Processor Fundamentals, IEEE Press,1997
[11] 曹志刚,钱亚生．现代通信原理．北京：清华大学出版社,2000
[12] Milola J. The Challenges in the Globalization of Software Radio. IEEE Communications Magazine,1999 (5): 84－89
[13] Jeffery A W. Analog－to－Digital Converters and Their Applications in Radio Receivers. IEEE Communications Magazine. 1995(5): 39－45
[14] 李瀚荪．电路分析基础(上、中、下册)．北京：高等教育出版社,1997
[15] 吴麒．自动控制原理(上册)．北京：清华大学出版社,1990
[16] 张葛祥,李众立,毕效辉．三级倒立摆非线性模型的建立．西南工学院学报,2001,16(4): 5－10
[17] 戴忠达．自动控制理论基础．北京：清华大学出版社,1991
[18] 黄家英．自动控制原理(上、下册)．南京：东南大学出版社,1991
[19] 欧阳黎明．MATLAB 控制系统设计．北京：北京国防工业出版社,2001
[20] 冯冬青,崔玮,杨秀红．线性最优控制系统加权矩阵的仿真研究．郑州工业大学学报,2000,21(1): 11－14
[21] 强明辉,周鹏等．利用遗传算法优化线性二次型调节器(LQR)．甘肃工业大学学报,1998,24(4): 51－55
[22] 肖建．现代控制系统综合与设计．北京：中国铁道出版社,2000